Sistemas Informáticos
Curso 08-09

Mountain Bytes:
Sistema de Control y Comunicación Inalámbrica de Redes Sensoriales Aplicado al Ejercicio

Helena Lorenzo Granizo

Javier Murillo Yagüe

Alberto Velázquez Alonso

Dirigido por:

Prof. Luis Hernández Yáñez

Departamento de Sistemas Informáticos y Programación

Facultad de Informática

Universidad Complutense de Madrid

Índice de contenido

Resumen / Summary

Este proyecto es una plataforma que integra hardware y software con el objetivo de comunicar, gestionar y recoger datos de dispositivos físicos, persistir estos datos y mostrarlos en diversos formatos amigables para el usuario.

Entre sus usos prácticos están el control de maquinaria industrial, electrodomésticos o aplicaciones domóticas. En nuestro caso hemos querido demostrar su funcionalidad implementando la gestión de las máquinas y usuarios de un gimnasio, desde el control de los usuarios y sus planes de trabajo personalizados hasta el envío de órdenes a la máquina de cada usuario para el control del ejercicio.

Este proyecto cubre todo el espectro de la informática, combinando una aplicación de calidad industrial accesible vía web que utiliza bases de datos para persistir información, con la programación de un microcontrolador, la interconexión de componentes hardware y la comunicación inalámbrica entre nodos.

This project is a platform which integrates hardware and software with the aim of communicating, managing and gathering data from physical devices, persisting this data and displaying it in different user-friendly formats.

Among its uses are the control of industrial machinery, appliances or home automation devices. In this case we have intended to show its functionality by implementing a management solution for a gym, from the monitoring of their users to the communicaton with the different machines to control their workout plans.

This project encompasses the whole spectrum of computer sciences, combining the development of an enterprise-grade web application which persists its data on a database with programming microcontrollers, interconnecting devices and wireless communtication between nodes.

Estado del arte – Innovación en el proyecto

Los orígenes de este desarrollo se encuentran en dos ideas complementarias sobre control de dispositivos y comunicación entre los mismos.

Por un lado, el control de cualquier tipo de máquina o dispositivo por computador no es una idea nueva, pero el abaratamiento de la tecnología ha permitido que este control se realice con computadores cada vez más pequeños, potentes y fáciles de programar, hasta el punto de que puede realizarse buena parte del procesamiento de los datos en el mismo dispositivo que los muestrea, en lugar de enviarlos a un computador central.

Desde el siglo XVIII en que se inventaron rudimentarios mecanismos para controlar telares y máquinas de vapor, la industria ha requerido constantemente de mejores sistemas de control para mejorar el rendimiento y la producción. Los primeros computadores fuera del ámbito militar se instalaron en plantas de producción, y hoy día casi todas las máquinas, incluso en el entorno doméstico, están controladas por una forma u otra de computador.

En la historia hay numerosos ejemplos de sistemas control mecánico, como el regulador centrífugo para la máquina de vapor, los sistemas de control de vuelo de los primeros aviones e incluso el termostato. En los años 40 comenzaron a aparecer los primeros computadores y en seguida comenzaron a tener aplicaciones en el control, especialmente en el campo militar y aeroespacial. Podemos citar el control de tiro en los barcos y el guiado de misiles, el control de vuelo más sofisticado en aviones denominado aviónica, o todos los sistemas de control para naves y satélites usados en los programas espaciales de Rusia y EEUU.

Sin embargo no fue hasta la década de 1970 en que empezaron a aparecer los primeros microcontroladores, pequeños dispositivos más baratos que un computador y que permitían algún tipo de programación. Un microcontrolador no es más que un circuito integrado que contiene un sencillo computador, normalmente más pequeño y especialmente más sencillo

que los computadores del mismo período de tiempo. La simplicidad implica menores costes en diseño, fabricación y consumo energético, ya que cuando aparecieron los microcontroladores, el coste de un computador era mucho más elevado que actualmente.

Hoy día el coste de estos sistemas de control ya no es un problema, puesto que las arquitecturas que se utilizan apenas repercuten en el precio total del producto. Su principal inconveniente es la poca flexibilidad que ofrecen, tanto en el rango de dispositivos o máquinas que pueden controlar, como en la dificultad de modificar el programa con el que funcionan para añadir nuevas funcionalidades o para adaptarse a los cambios que los fabricantes suelen incluir en las máquinas que deben controlar.

Si bien los primeros aparatos que llevaron microcontroladores fueron los equipos de sonido y vídeo, hoy día se utilizan para un elevado número de aplicaciones tanto en el entorno doméstico como en el industrial. Los ejemplos más relevantes son: acondicionamiento de aire y calefacciones, audio y vídeo, sistemas de seguridad, dispositivos de intercomunicación, control de iluminación, y por supuesto maquinaria industrial y robots.

Actualmente se están desarrollando nuevas soluciones con el objetivo de aumentar la flexibilidad y reducir la complejidad de desarrollar nuevo código, que supone poder gestionar dispositivos que antes no resultaba rentable conectar a un controlador.

Una de estas soluciones es la que vamos a utilizar en nuestro proyecto: Arduino es la combinación de un microcontrolador y placa base asociada con un entorno de desarrollo que simplifica la programación de este controlador y que por tanto permite la comunicación con un gran número de dispositivos muy diferentes con un esfuerzo de programación mucho menor que si hubiera que programar el microcontrolador a mano.

La alternativa más evidente al uso de Arduino es precisamente el desarrollo de nuestra propia placa. Esto tiene la ventaja de un menor coste en materiales, ya que la adquisición de uno o varios microcontroladores es más barato que una placa Arduino completa, pero implicaría por un lado la programación en C o lenguaje ensamblador de la arquitectura del microcontrolador de todo el código que nosotros hemos desarrollado en el IDE de Arduino.

Por otro lado, implicaría también el diseño y construcción de nuestras propias placas, tanto de Arduino como de ZigBee, así como su interconexión y depuración, lo que implicaría mayores costes no sólo económicos sino especialmente en tiempo hasta desarrollar un producto con una funcionalidad comparable a Arduino.

Existen además otras placas comerciales con funcionalidades parecidas a Arduino, como por ejemplo PowerJaguar o Sanguino. Las ventajas de Arduino sobre sus competidores es que al estar mucho más extendido es más sencillo encontrar documentación sobre su funcionamiento y componentes de reemplazo si fuera necesario, además de tener un mercado mucho más extenso de componentes y sensores que permitirían añadir nuevas funcionalidades si existiese la necesidad. Además es de interés destacar que debido al éxito de Arduino están surgiendo iniciativas como FreeDuino que pretenden ser placas compatibles con Arduino, permitiendo cambiar de proveedor si fuese necesario.

Pasando a la comunicación entre aparatos y controladores, el control de dispositivos siempre se ha hecho mediante cableado, que supone importantes desventajas. Enumerando sólo algunas, tenemos que la instalación del cableado es cara y poco flexible, además de difícil de reparar en caso de que se deteriore. Obliga además a mantener los dispositivos que se desean controlar relativamente cerca del computador que los controla, e impiden que en caso de necesidad estos dispositivos sean desplazados de su sitio.

Sin embargo están apareciendo nuevos métodos de transferencia de información sin cables que están cambiando este panorama. Primero fueron los infrarrojos, que no se utilizan demasiado fuera del entorno doméstico porque necesitan visión directa entre los dispositivos que se están comunicando. Posteriormente se empezaron a usar sistemas de radiofrecuencia como WiFi, que permiten una flexibilidad sin precedentes pero que consumen mucha energía y por tanto son poco idóneos para dispositivos de bajo consumo o que no tienen una fuente de alimentación cercana.

Hace unos años apareció una nueva generación de dispositivos inalámbricos por radiofrecuencia, Bluetooth, muy utilizada actualmente en entornos domésticos pero que aún se podía mejorar tanto en el ámbito del consumo como en el de la fiabilidad de la conexión.

La última tecnología desarrollada en el campo de la comunicación por radiofrecuencia es ZigBee. Se desarrolló desde un principio con el objetivo de ser más simple y barata que WiFi o Bluetooth, pero además de bajo consumo ofrece otras características muy interesantes como seguridad integrada en la transmisión, o "mesh networking": la posibilidad de que dos nodos cualesquiera de la red se comuniquen entre sí a través de otros nodos si no existe posibilidad de establecer una conexión directa, y que según la calidad de la red en cada instante pueda elegirse dinámicamente a través de qué nodos se establecerá la conexión.

ZigBee es la tecnología que hemos utilizado en nuestro proyecto, fundamentalmente por su menor consumo, pero también por las capacidades de mesh networking.

Para el computador central que recibirá las conexiones de todos los Arduino, hemos desarrollado una aplicación que gestiona la recepción y el envío de mensajes desde y hacia cada un o de os Arduino, pero además permite guardar registro de todas las comunicaciones y datos transferidos en una base de datos, que en nuestro caso es MySQL, y también la gestión vía web de esta apliación mediante JSP sobre Tomcat. La aplicación esta desarrollada sobre Spring, un framework que facilita la creación de aplicaciones MVC con interfaces web y uso de bases de datos. Los datos recogidos de los Arduino se representan de forma gráfica usando un applet Java o bien JavaScript+jQuery, y también en forma de informe usando JasperReports.

Objetivos

Desde un punto de vista general, nuestro objetivo es poder controlar un dispositivo físico, como por ejemplo una máquina, usando una placa Arduino con su correspondiente microcontrolador. Para ello, nuestra placa Arduino enviará datos relevantes de este dispositivo y recibirá instrucciones de una aplicación alojada en un servidor remoto. Esta aplicación alamacenará los datos recibidos y las instrucciones enviadas para consultas futuras. La comunicación se realizará de forma inalámbrica entre dos placas Arduino, una de ellas conectada al dispositivo físico y la otra conectada al servidor que aloja la aplicación. El diseño de la aplicación prevé poder controlar varios dispositivos físicos, cada uno conectado a su correspondiente placa Arduino.

Concretamente, en nuestro proyecto vamos a gestionar un único dispositivo físico, en este caso una bicicleta estática. Para ello habrá dos placas, una conectada a la bicicleta que contará las vueltas de la rueda de la bicicleta, y otra conectada al servidor que servirá de enlace para el envío y recepción de datos.

Podemos dividir los objetivos del proyecto en capas:

1. A nivel microcontrolador, el objetivo es conseguir que nuestras placas Arduino sean capaces de aceptar señales desde botones y teclado, así como de escribir texto en una pantalla LED.

2. A nivel de red, deseamos que varias placas Arduino se puedan conectar a un computador mediante USB, y además entre sí usando la tecnología ZigBee antes descrita.

3. Queremos que el programa que se ejecuta en el computador pueda conectarse y comunicarse con una placa Arduino mediante USB.

4. También pretendemos que la aplicación pueda guardar y recuperar datos de una base de datos.

5. Además, la aplicación deberá ser accesible mediante web, usando un navegador.

6. La aplicación debe ser capaz de gestionar diferentes parámetros en los diferentes nodos de la red. Específicamente, nuestra aplicación debe ser capaz de controlar diferentes tipos de entrenamiento en la bicicleta si así lo ha elegido el usuario.

Control de periféricos en Arduino

El objetivo en esta capa es controlar diferentes periféricos, tales como botones, un teclado numérico o un display LED usando el microcontrolador de Arduino.

Uno de los puntos problemáticos en la gestión de periféricos es el número de salidas disponibles. Para los periféricos indicados, lo natural es utilizar los pines de entrada/salida digital que ofrece la placa, pero no son suficientes para todos los dispositivos que tenemos que controlar, así que debemos hacer uso también de los pines analógicos.

El control de un display es también problemático, por la dificultad de gestionar la escritura de caracteres y por el hecho de que el display debe ser constantemente refrescado.

Conectividad en Arduino

Si bien la implementación del protocolo ZigBee se hace enteramente en hardware, nosotros tenemos que configurar cada nodo de la red para que efectúe la conexión de forma adecuada, ya que ZigBee admite varios tipos de red y no todos son compatibles entre sí.

Gestión de la base de datos

Queremos que nuestra plataforma sea capaz de leer datos de una red ZigBee, escribirlos en una base de datos y que estén disponibles posteriormente o a la vez para ser recuperados y opcionalmente representados gráficamente.

Interfaz web

La interfaz de nuestra aplicación deberá funcionar vía web. El usuario podrá ver sus progresos y diferentes sesiones de entrenamiento mediante un navegador, conectándose al servidor de la aplicación. El usuario podrá también gestionar diferentes entrenamientos personalizados. El administrador de la aplicación deberá ser capaz de acceder y modificar todos los aspectos de las diferentes máquinas que gestiona esta aplicación: usuarios, tipos de máquinas y entrenamientos.

Gestión de la actividad en las máquinas

Tanto los usuarios como el administrador deben poder especificar entrenamientos específicos para cada usuario, y la aplicación debe ser capaz de seguir el progreso de este entrenamiento en la máquina, así como controlar diferentes aspectos del mismo como por ejemplo el nivel de esfuerzo de la máquina.

Descripción del proyecto

Visión general

El objetivo de este proyecto es establecer una red inalámbrica en la que un cierto número de máquinas, en este caso bicicletas de gimnasio, se comuniquen con una aplicación en nuestro servidor central que salvará los datos recibidos sobre la actividad de los usuarios, enviará órdenes a las máquinas para modificar diferentes parámetros como el esfuerzo, y permitirá a los usuarios visualizar los datos recogidos de diferentes formas así como crear y modificar diferentes sesiones de entrenamiento. En nuestro proyecto vamos a controlar sólo una de estas bicicletas, pero el diseño del conjunto permitiría añadir más máquinas sin modificar más que la configuración de la red inalámbrica.

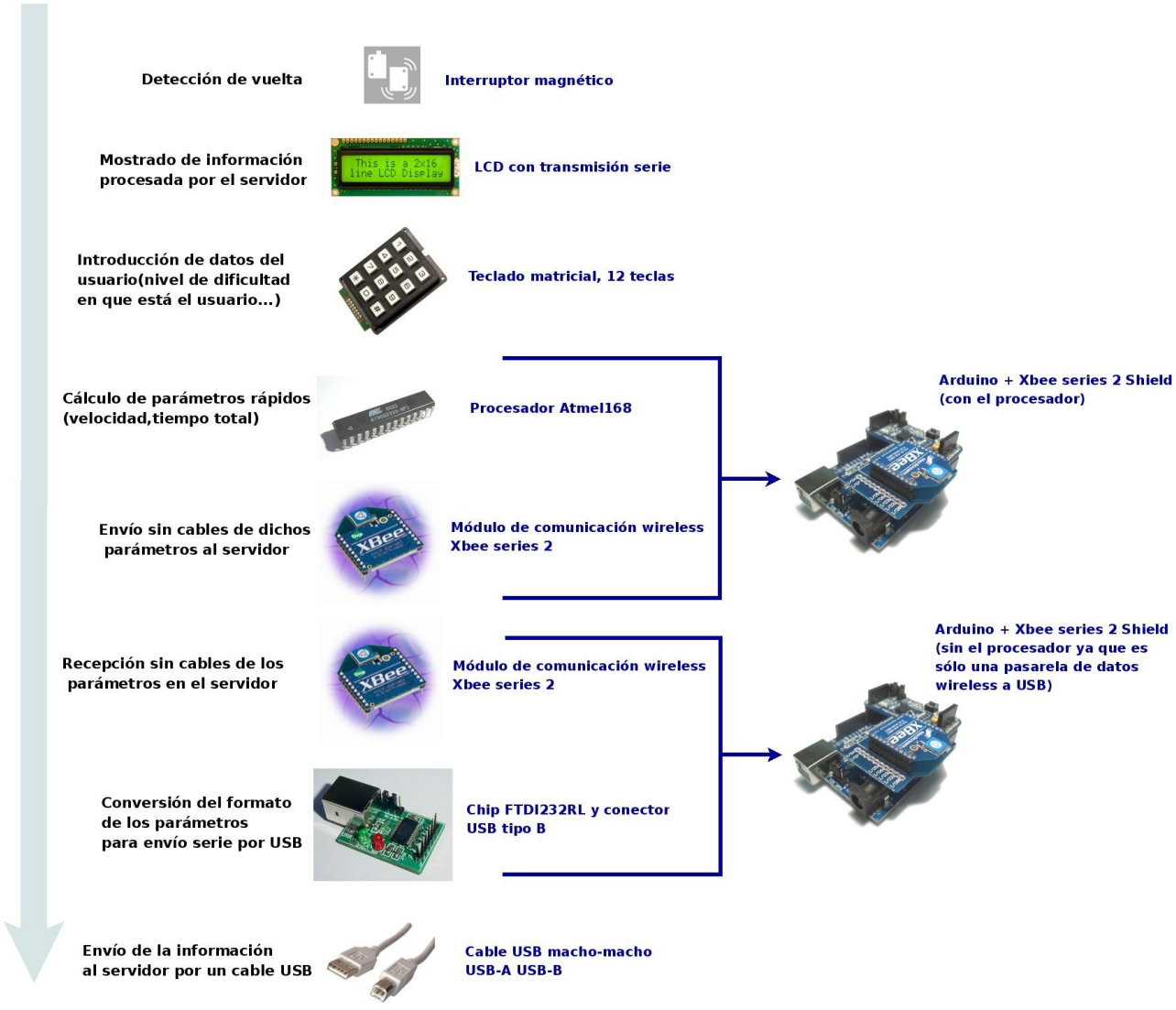
Este proyecto consta de dos grandes apartados: Por un lado, una parte relacionada con la placa Arduino y la comunicación mediante ZigBee. Por otro lado, una aplicación que recibe los datos de estas placas, los guarda en una base de datos y permite su recuperación vía web, así como el envío de instrucciones a las placas.

La placa Arduino incorpora puertos tanto digitales como analógicos para interactuar con dispositivos externos. En este proyecto, esta placa gestiona cuatro botones direccionales, dos más para aceptar o cancelar, un teclado numérico completo para hacer login e introducción por parte del usuario de otros datos, y por supuesto la conexión con la máquina que vamos a controlar, que es una bicicleta estática. Esta conexión consta de un imán que envía una señal por el cable hacia la placa cada vez que la rueda de la bicicleta da exactamente una vuelta.

La placa Arduino incorpora también un temporizador interno que es sincronizado con la hora del computador central cuando se carga el programa. Este temporizador se usa después para marcar cada mensaje que se envía al computador con el instante en el que ocurrió, tratando de mitigar el posible retraso en la red.

Otro dispositivo importante que va conectado a la placa Arduino es el "XBee shield", una placa auxiliar que implementa el protocolo de comunicación por radiofrecuencia ZigBee y permite a esta placa comunicarse con las demás.

HARDWARE NECESARIO PARA LA IMPLEMENTACIÓN



Toda la programación de las comunicaciones y de la lectura de datos se realiza en Processing/Wiring, un entorno de programación que permite al desarrollador no tener que usar C o ensamblador para desarrollar código para el microcontrolador.

En el lado de la aplicación, se utiliza un framework llamado Spring, para el lenguaje de programación Java. Este framework usa a su vez una biblioteca específica para la

persistencia de datos denominada Hibernate, que permite crear una correspondencia entre objetos de Java y tablas de la base de datos. En concreto, la base de datos que utilizaremos es MySQL.

Del lado del usuario, este framework provee la capacidad de acceder a nuestra aplicación mediante web a través de JSPs, componentes Java que permiten la interacción entre objetos de Java y páginas web. Esta tecnología permite también generar páginas web dinámicas en respuesta a las peticiones del usuario. Para ello necesitamos un servidor web que implemente JSP, y nuestra elección es Tomcat versión 5.5 o superior.

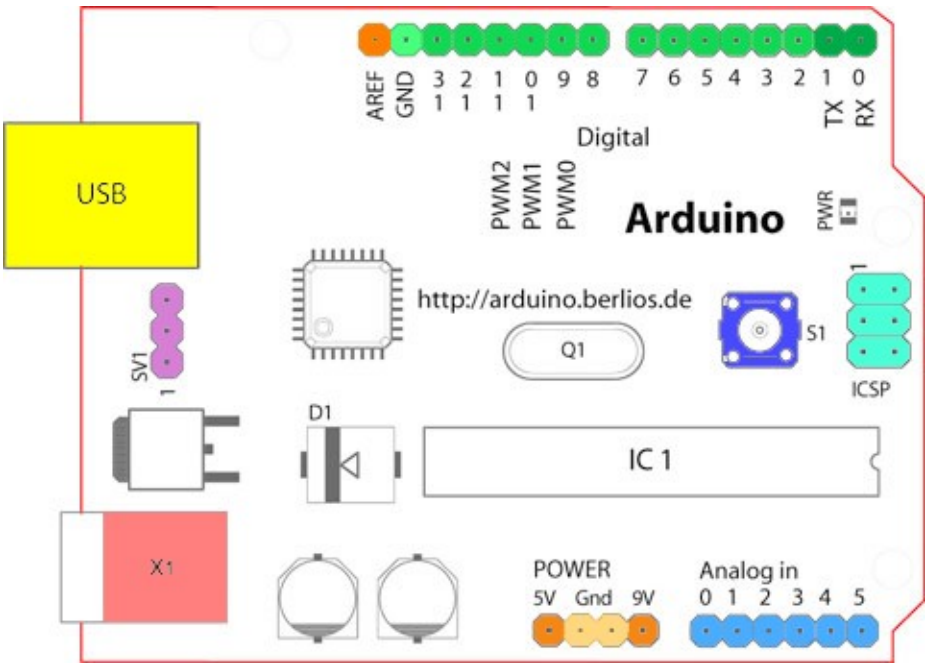
En la parte de visualización de datos e interacción con el usuario, tenemos un applet Java que utiliza una biblioteca de representación gráfica, OpenChart2, para mostrar visualmente los datos almacenados en la base de datos. Opcionalmente, este applet permite visualizar la actividad en la bicicleta casi en tiempo real.

Disponemos también de un gráfico generado dinámicamente usando JavaScript y jQuery que permite no sólo representar la actividad del usuario sino modificar las tablas de entrenamiento pinchando directamente sobre él.

Arduino

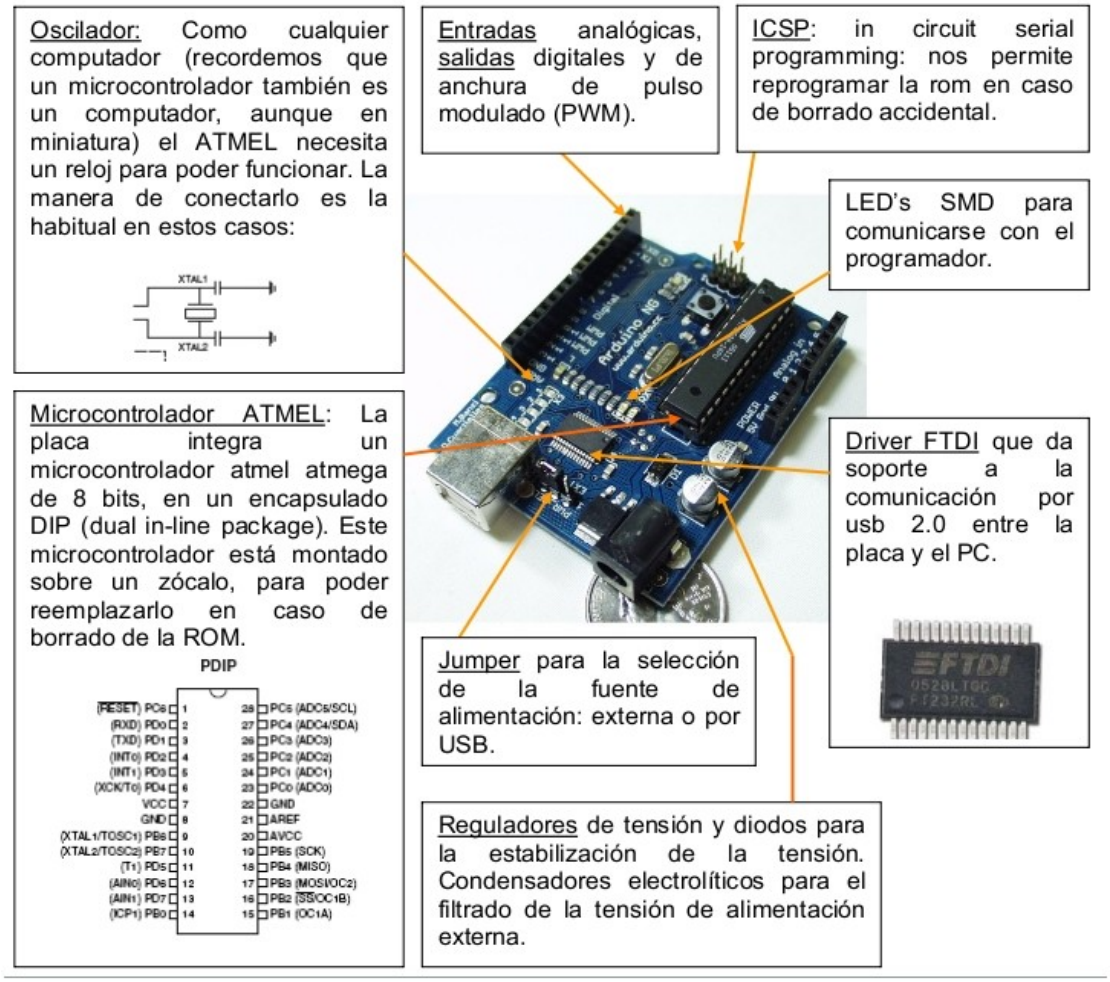
Arduino es una plataforma de desarrollo compuesta por una placa I/O cuyo diseño está bajo licencia libre, y por un entorno de desarrollo basado en Processing/Wiring.

Existen varias versiones de placas Arduino; pero todas incorporan un microcontrolador Atmel y una conexión RS-232 para permitir la transferencia del programa que se va a ejecutar, y para poder comunicarse con un computador. La placa Arduino que vamos a utilizar incorpora un microcontrolador Atmega168 a 16 MHz y 8Kb de memoria interna. Algunos modelos, como el que utilizamos en este proyecto, implementan esta conexión mediante USB a través de un chip conversor FTDI.



Descripción de la placa Arduino siguiendo las agujas del reloj:

- Pin de referencia analógica (naranja)
- Señal de tierra digital (verde claro)
- Pines digitales 3-13 (verde)
- Pines digitales 1-2 / entrada y salida del puerto serie: TX/RX (verde oscuro)
- Botón de reset (azul oscuro)
- Entrada del circuito del programador serie (azul turquesa)
- Pines de entrada analógica 0-5 (azul claro)
- Pines de alimentación y tierra (fuerza: naranja, tierra: naranja claro)
- Entrada de la fuente de alimentación externa (9-12V DC) – X1 (rosa)
- Conmuta entre fuente de alimentación externa o alimentación a través del puerto USB – SV1 (violeta)
- Puerto USB (amarillo)



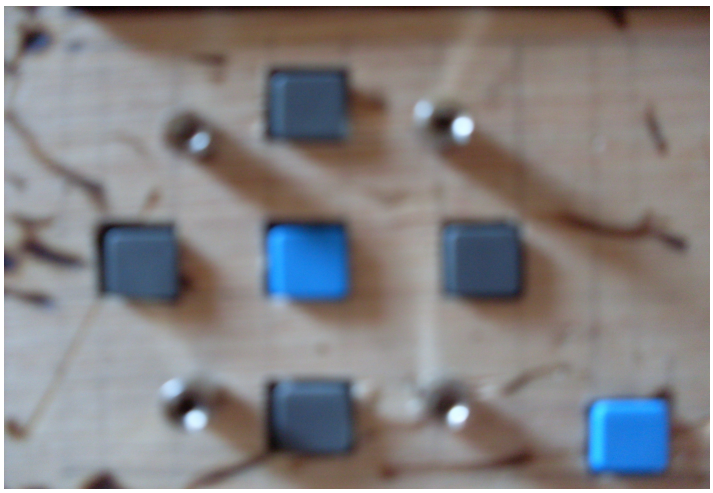
La placa base admite la conexión con módulos hardware adicionales denominados "shields" que añaden funcionalidad a la placa. Por ejemplo existen shields que permiten la conexión WiFi o Bluetooth, o como en el caso se nuestro proyecto, conexión mediante ZigBee.

La plataforma Arduino incluye además un entorno de desarrollo que permite escribir programas usando un lenguaje denominado Processing en lugar de tener que escribirlos en ensamblador.

Este entorno hace uso de una biblioteca llamada Wiring, que simplifica las operaciones de entrada y salida.

Gestión de botones en Arduino

Nuestro hardware cuenta con 6 pulsadores físicos con los que el usuario puede interactuar: arriba, abajo, izquierda, derecha, centro y cancelar.



Al implementar esta funcionalidad (que principalmente se usa para que el usuario de la bicicleta introduzca el nivel en el que se encuentra en cada momento del entrenamiento), nos encontramos con varias limitaciones:

La primera de ellas y más importante, fue que nos quedamos sin pines digitales de lectura en la placa de arduino, esto significaba que no podríamos leer botones digitales. Pero la placa cuenta con 6 pines de lectura analógica, esto es, seis pines donde el procesador de arduino muestrea valores de potencial, hace una conversión analógico-digital y devuelve un valor al procesador de entre 0 y 255. La solución estaba clara, se optó por leer los valores de dichos pines y si el valor leído era superior a 255/2 (128) entonces se consideraba como pulsación, si no, el botón no se había pulsado.

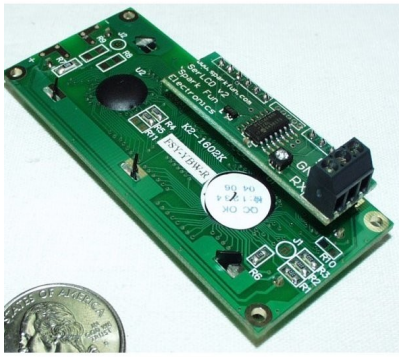
La segunda complicación consistía en que se deben gestionar las colisiones entre los diferentes eventos que hacen bloquear al procesador, es decir, eventos que hacen que el procesador se quede esperando hasta su finalización. En este caso, cuando el usuario pulsa un botón, el procesador debe esperar a que el usuario deje de pulsarlo, ya que si no es así se detectarán continuamente pulsaciones durante el tiempo que el usuario pulse. Se ha tenido que gestionar esta situación para evitar precisamente que la pulsación de un botón impidiera que se detectaran paso de vueltas de la bicicleta.

Gestión del display en Arduino

El display consiste en un LCD al que se le envían las órdenes de visualización vía serie. Para lograr esta funcionalidad, este hardware consta de dos partes bien diferenciadas:

- Pantalla LCD (con 16 pines que controlan cada uno de los 2x16 leds con los que cuenta esta pantalla LCD)

- Microprocesador HD44780.(Se puede ver en la parte de atrás, segunda imagen adjunta).



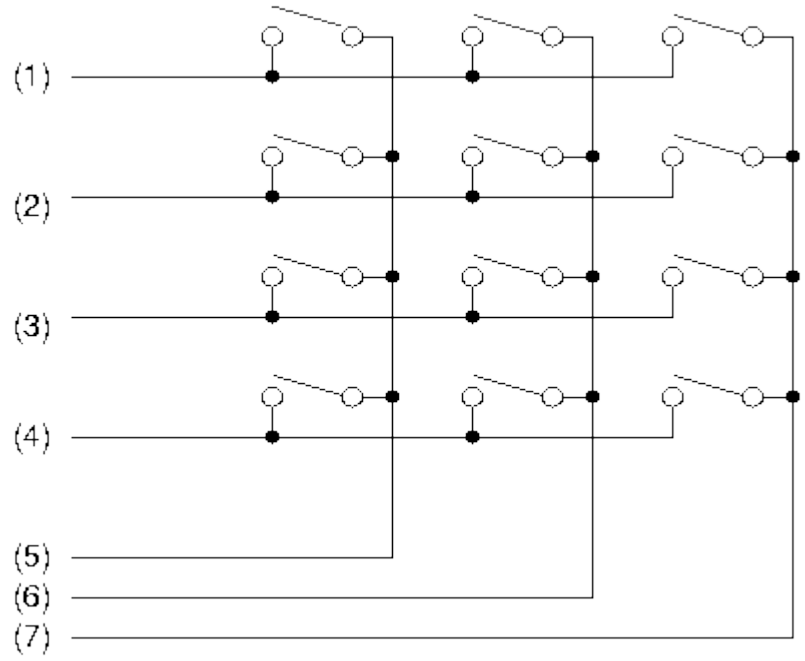
Este microprocesador recibe cadenas de caracteres en serie a 9600 bps y las traduce a órdenes directas para la pantalla LCD, haciendo infinitamente más sencilla la ardua tarea de trabajar con uno de estos dispositivos.
Además, y mucho más importante, es que permite que integremos una pantalla LCD con el coste de un sólo pin de nuestra placa, y no 16 pines como hubiéramos tenido que usar sin el paso intermedio de este microprocesador.

Aun así hemos tenido serias complicaciones para integrar este componente a nuestra placa.
Esta placa cuenta únicamente con un UART, es decir, con un solo pin de comunicación serie para la transmisión de datos, y otro pin de comunicación serie para la recepción de los mismos. Ambos pines están ligados a nuestra placa de comunicación wireless Xbee y están ocupados. Para solucionar este problema usamos simulación Software (USART).

Aunque un puerto común de Arduino no disponga de los módulos hardware necesarios para la comunicación serie, podemos, mediante software, simular un comportamiento de envío y recepción de datos por ese pin digital. Es cierto que esto sobrecarga el procesador de nuestra placa, ya que forzamos a nuestro procesador a que continuamente esté vigilando este pin, pero es la única solución posible.

Teclado matricial

El teclado matricial no son más que 12 interruptores interconectados de la manera que se explica en la imagen a continuación:

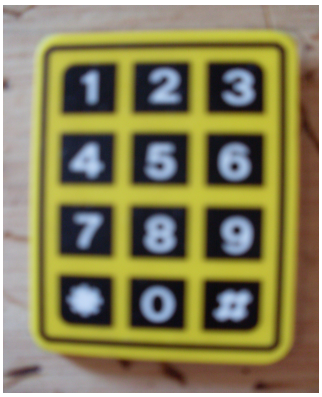


Se puede apreciar que aunque son 12 botones, sólo hay 7 líneas de control con las que interactuaremos. Para saber qué botón ha sido pulsado debemos estar continuamente "preguntando" al dispositivo como se explica a continuación.

Tendremos dos bucles anidados. Para cada una de las líneas del 1-4 introduciremos un "1" lógico, de manera exclusiva, es decir, primero introduciremos un 1 en el pin 1, y 0 en los pines 2,3,4.

A continuación miraremos los valores de los pines 5,6 y 7 si alguno de ellos es "1" significará que el circuito ha sido cerrado, (el botón ha sido pulsado). Dependiendo de si ha sido el pin 5 ,6 o 7 identificaremos el botón.

De nuevo tendremos al procesador continuamente evaluando los valores y cambios de estos pines.



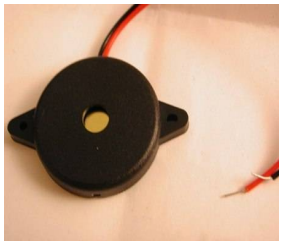
Beeper

Se decidió introducir un dispositivo de aviso sonoro por múltiples motivos:

- Es necesario que el usuario de la bicicleta se percate de que debe realizar alguna modificación durante su entrenamiento, como por ejemplo subir o bajar la resistencia, aumentar o disminuir la velocidad...

- Es necesario que el usuario sepa cuando ha pulsado un botón ya que si no puede realizar varias pulsaciones pensando que sólo hace una.

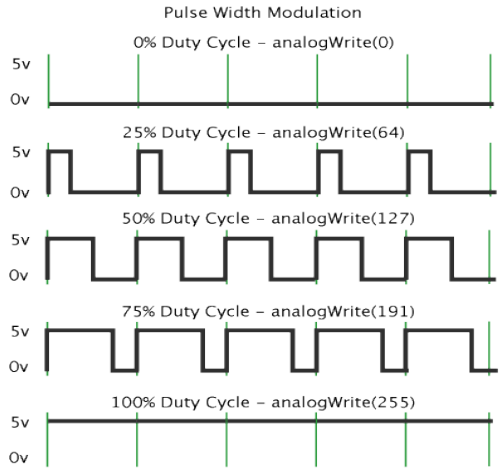
- Si hay algún problema con el dispositivo, es una manera de avisar a algún encargado de que algo no va bien.



Este hardware traduce voltajes de corriente continua en un pitido más o menos agudo dependiendo del valor del potencial introducido en ambos de sus polos.

Como deseamos tener varios tonos de aviso con diferentes frecuencias necesitamos introducir dependiendo del momento que queramos un potencial u otro.

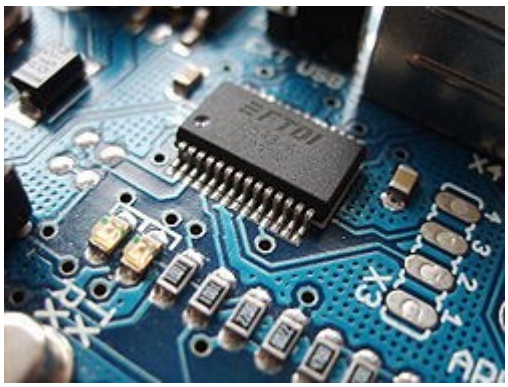
Arduino cuenta con 6 pines digitales en los que no sólo podemos asignar un 1 o un 0 (5v o 0v) sino valores de voltaje variables de cero a 255. Son Pines PWM, es decir, Pulse Width Modulation. En otras palabras, no siempre asignamos 5v al pin, sino que alternamos entre 5v y 0v y de esta manera "simulamos" un valor de voltaje inferior a 5v. Es una manera de obtener resultados analógicos con pines digitales. En la siguiente ilustración queda bastante clara esta tecnología:



Es de este modo como logramos introducir valores de entre 0 a 255 y haremos "vibrar" nuestro beeper a la frecuencia deseada, consiguiendo diferentes sonidos para nuestras diferentes señales de alerta.

Comunicación USB entre la placa y el computador

La comunicación USB entre arduino y el computador se realiza gracias al chip conversor FTDI:



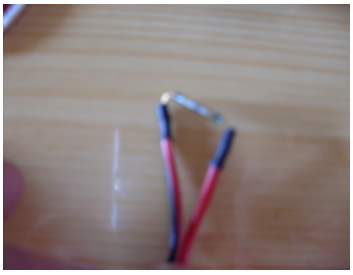
Este chip se encarga de la conversión del protocolo serie RS-232 a la tecnología USB, haciendo de manera muy transparente la comunicación de nuestro PC con nuestro procesador ATMEL168, que sólo gestiona el FIFO de comunicaciones, y las colisiones de mensajes.

Este chip permite comunicaciones serie con un amplio abanico de velocidades: 1200, 2400, 4800, 9600, 19200, 38400, 57600 y 115200 bps.

Nosotros hemos usado una velocidad de conexión de 9600 baudios para garantizar una buena seguridad en las transmisiones, aunque se han probado velocidades mayores y no ha habido problemas, consideramos más fiable esta velocidad.

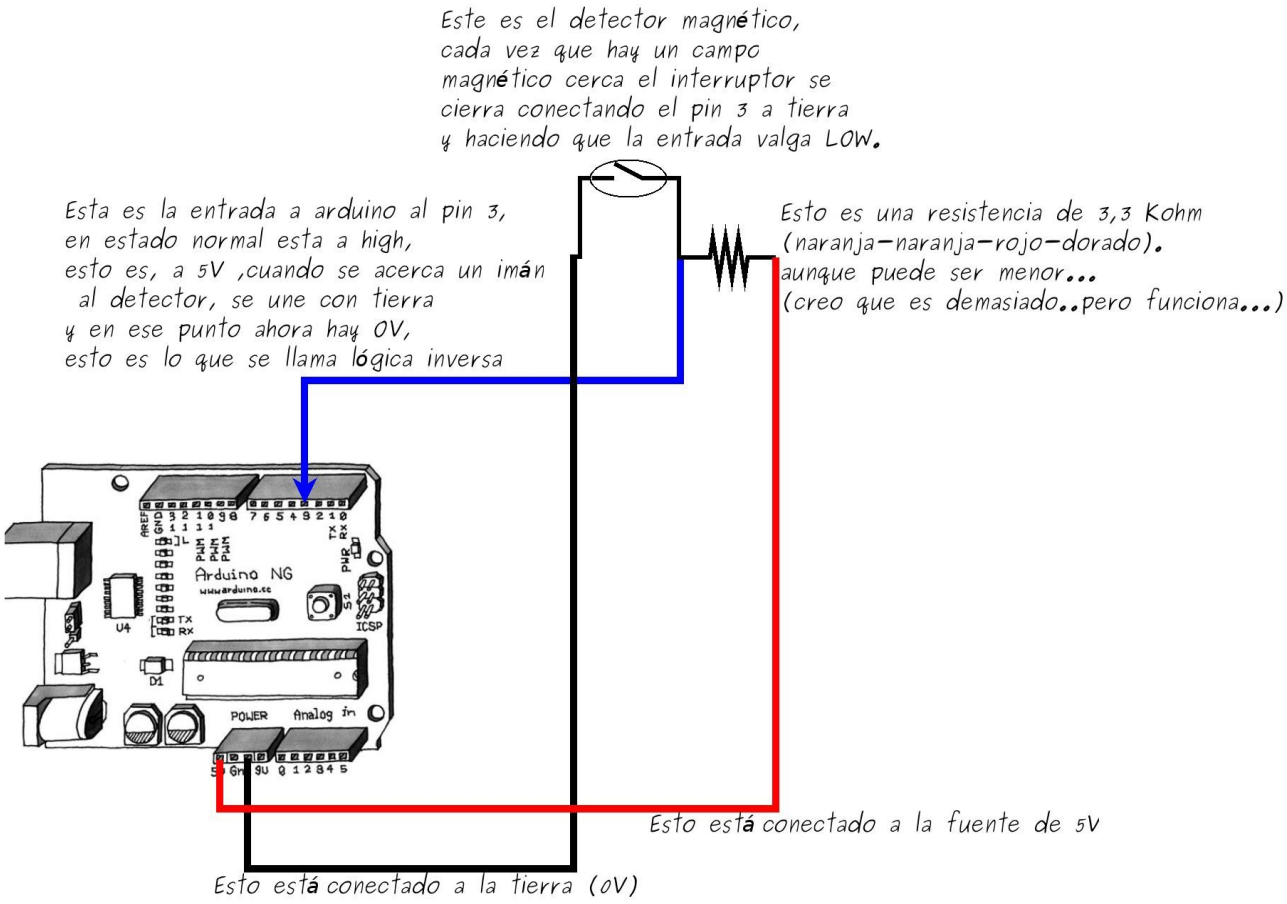
Detector de vueltas

Se ha decidido implementar el detector de vueltas con un sensor magnético, tal y como se hace en el mundo comercial con este tipo de productos. Este dispositivo es simplemente un interruptor que se cierra al paso de un material ferromagnético, como un imán por ejemplo.



Cuando un imán se acerca al componente, éste cierra un circuito que pondrá el pin al que esté conectado a 5 voltios (o a cero, dependiendo de la lógica que hayamos elegido).

El esquemático del circuito sería así:



ZigBee / XBee

En la última década se han ido desarrollando estándares de comunicación inalámbrica como WiFi o Bluetooth con el objetivo de incorporar más características a la vez que se reduce el consumo. Pero para algunos usos, estos estándares siguen necesitando demasiada potencia para funcionar, y éste es el origen de ZigBee.

El objetivo fundamental de ZigBee es permitir la comunicación mediante radiofrecuencia reduciendo todo lo posible el consumo. Sin embargo posee otras características que lo hacen

muy interesante como alternativa a otros protocolos incluso si el consumo no es la prioridad.

Una de estas características es la denominada “mesh networking”, cuyo principio de funcionamiento es que dos nodos cualesquiera de una red no necesitan comunicarse directamente uno con otro sino que lo pueden hacer a través de otros nodos de la misma red. Este modo de funcionamiento es muy útil si las ondas de radio no llegan con la calidad suficiente entre dos nodos.

XBee es una implementación de ZigBee para Arduino.

Comunicación entre el Xbee Shield y Arduino

El módulo de comunicación ZigBee para Arduino permite comunicación inalámbrica en un rango de hasta 30 metros si existe visión directa entre dispositivos, es decir, si no hay paredes que obstaculicen la comunicación.

Antes de iniciar la comunicación entre dispositivos, cada uno de ellos debe ser configurado. Los parámetros que deben introducirse son: el identificador del dispositivo; el identificador y canal de radiofrecuencia que va a utilizar la red completa, así como los dispositivos vecinos a los que cada uno de los dispositivos puede conectarse. Puede tambien configurarse el dispositivo para que se conecte a cualquier otro dispositivo de su red.

Comunicación entre el XBee Shield y el computador

Una interesante característica del Xbee Shield es que la placa en la que va montado no necesita llevar microcontrolador si no es necesario procesar los datos. Es decir, que la placa que va conectada al computador no necesita llevar microcontrolador porque los datos se transmiten directamente del XBee Shield desde y hasta el computador mediante el cable USB.

El computador recibe y envía datos a los demás dispositivos de la red inalámbrica a través del XBee Shield; se comunica con el Shield a través de USB mediante un emulador de terminal que simplemente envía texto plano. Este texto pueden ser datos o bien comandos para dar órdenes al XBee; este tipo de información se denomina comandos AT. Ya hemos detallado previamente cómo se comunica el computador con la placa a través de USB.

El computador recibe de la red texto plano que es procesado por la apliacación encargada de gestionar las máquinas. Esta apliacaión también envía ocasionalmente texto, con el fin de solicitar información de las placas Arduino conectadas a cada una de las máquinas que deseamos controlar.

Comunicación entre nodos ZigBee

ZigBee está basado en el estándar IEEE 802.15.4-2006, que especifica las capas física y MAC. ZigBee especifica tres tipos de nodos de red: coordinador, router, y dispositivo final.

El coordinador es único, es la raíz de la red, y permite incluso la interconexión a través de él a otras redes. Los routers permiten reenviar datos de unos nodos a otros. Los dispositivos finales sólo envían información a un router o coordinador, y pueden estar la mayor parte del tiempo en standby, lo que implica un bajo consumo de energía.

Las comunicaciones Zigbee se realizan en la banda libre de 2.4GHz. A diferencia de bluetooth, este protocolo no utiliza FHSS (Frequency hopping), sino que realiza las comunicaciones a través de una única frecuencia, es decir, de un canal.

Normalmente puede escogerse un canal de entre 16 posibles. El alcance depende de la potencia de transmisión del dispositivo así como también del tipo de antenas utilizadas (cerámicas, dipolos, etc).

El alcance normal con antena dipolo en línea vista es de aproximadamente (tomando como ejemplo el caso de MaxStream, en la versión de 1mW de potencia)

de 100m y en interiores de unos 30m. La velocidad de transmisión de datos de una red Zigbee es de hasta 256kbps.

Una red Zigbee la puede estar formada teóricamente por hasta 65535 equipos, es decir, el protocolo está preparado para poder controlar, en la misma red, esta gran cantidad de dispositivos.

El uso del protocolo Zigbee va desde reemplazar un cable por una comunicación serial inalámbrica, hasta el desarrollo de configuraciones punto a punto, multipunto, peer-to-peer (todos los nodos conectados entre sí) o redes complejas de sensores. Una conexión típica suele ser que cada módulo Xbee posea algún tipo de sensor, el cual entrega los datos para ser enviados a través de la red a un Centro que administre la información.

Una red Zigbee la forman básicamente 3 tipos de elementos. Un único dispositivo Coordinador, dispositivos Routers y dispositivos finales (end points).

El Coordinador.

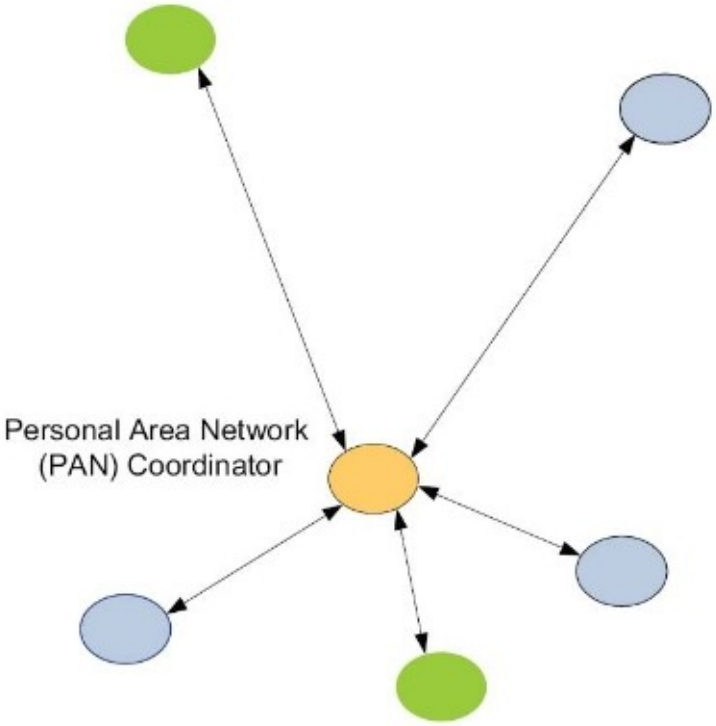
Es el nodo de la red que tiene la única función de formar una red. Es el responsable de establecer el canal de comunicaciones y del PAN ID (identificador de red) para toda la red. Una vez establecidos estos parámetros, el Coordinador puede formar una red, permitiendo unirse a él a dispositivos Routers y End Points. Una vez formada la red, el Coordinador hace las funciones de Router, esto es, participar en el enrutado de paquetes y ser origen y/o destinatario de información.

Los Routers.

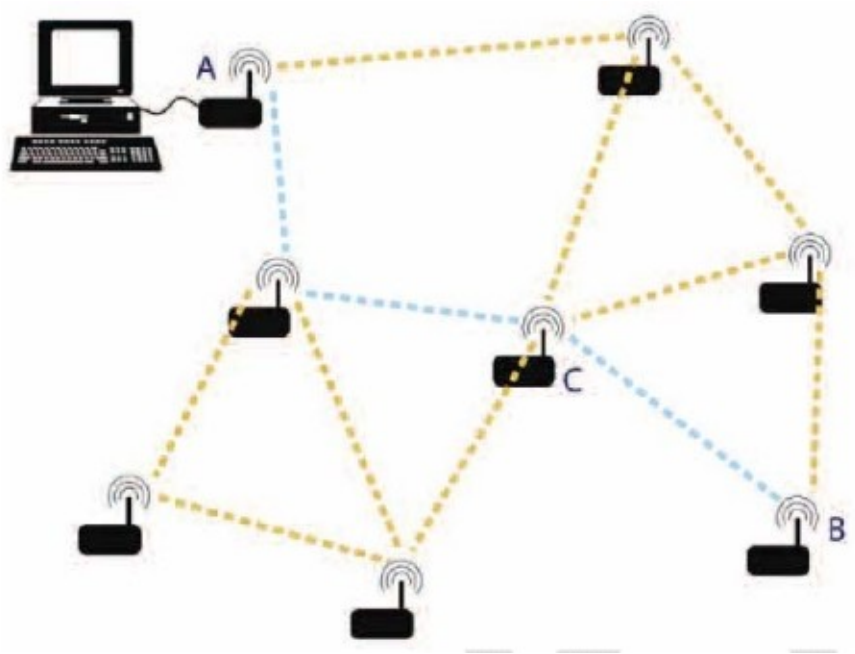
Es un nodo que crea y mantiene información sobre la red para determinar la mejor ruta para transmitir un paquete de información. Lógicamente un router debe unirse a una red Zigbee antes de poder actuar como Router retransmitiendo paquetes de otros routers o de End points.

End Device.

Los dispositivos finales no tienen capacidad de enrutar paquetes. Deben interactuar siempre a través de su nodo padre, ya sea este un Coordinador o un Router, es decir, no puede enviar información directamente a otro end device. Normalmente estos equipos van alimentados a baterías. El consumo es menor al no tener que realizar funciones de enrutamiento.

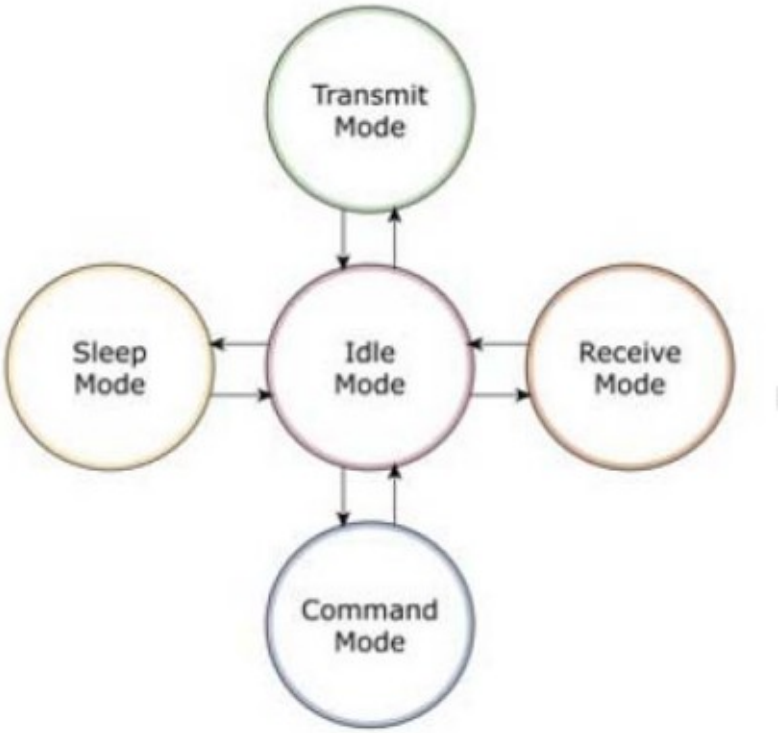


En este gráfico se aprecia en color naranja la configuración de Coordinador y en azul y verde los router y end device.



Ejemplo de una red Mesh de Series 2 que permite enviar mensajes de punto a punto pasando por diferentes XBee modules con diferentes configuraciones.

Los módulos XBee pueden operar en cinco modos diferentes que se indican a continuación:



Mesh networking

Este término hace referencia a un método de comunicación entre nodos de una red cuya característica más prominente es que la ruta que siguen los datos entre dos nodos cualesquiera no es fija, sino que puede variar por circunstancias como la carga de la red, la calidad de la conexión entre diferentes nodos, o el hecho de que algunos de los nodos puedan no estar operativos. Ante estos hechos, la red es capaz de modificar dinámicamente el proceso de enrutamiento haciendo que otros nodos sirvan como puntos intermedios para la transmisión. De esta forma, cada paquete de datos puede ser transmitido directamente al nodo de destino, o puede "saltar" a través de uno o varios nodos de la red hasta llegar a su destino.

Una de las aplicaciones fundamentales de este concepto es que una red puede "repararse" automáticamente: si uno de los nodos queda inoperativo o su conexión se colapsa, el resto de nodos se reconfiguran dinámicamente para evitar este nodo inactivo. Esta forma de funcionamiento añade un plus de fiabilidad a las redes inalámbricas, que por su modo de funcionamiento y medio físico de transmisión no son, en principio, tan fiables como una red cableada. Sin embargo esta forma de enrutamiento puede aplicarse también a redes cableadas, ya que es independiente del medio.

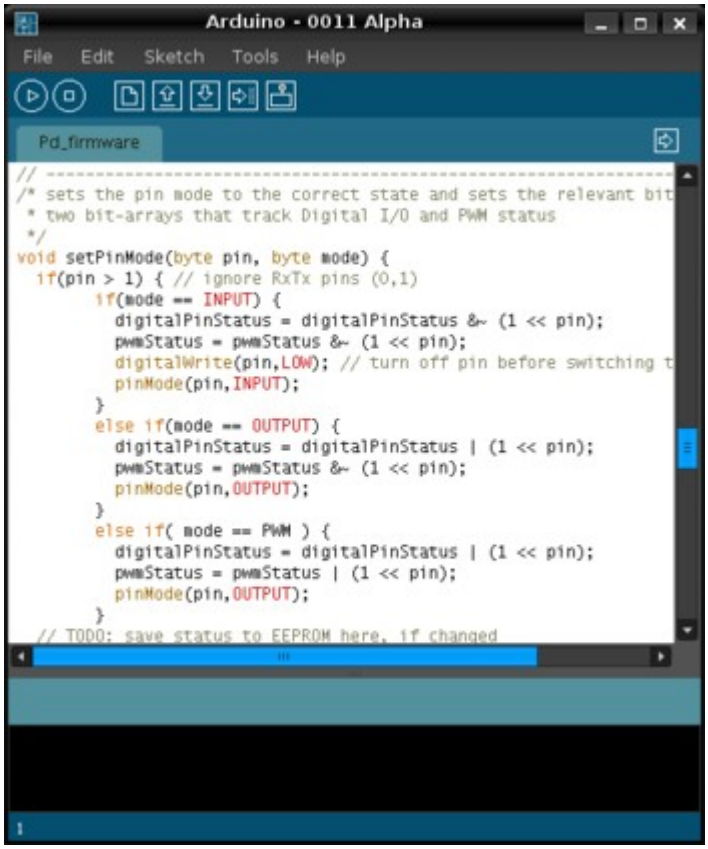
Otra característica es que este tipo de enrutamiento permite cambiar la localización física de los nodos sin necesidad de reconfigurar la red, siempre que la nueva localización esté dentro del rango de algún otro nodo configurado para reenviar datos. Esta capacidad implica un menor mantenimiento de la red, especialmente en el supuesto de que los nodos sean susceptibles de ser movidos con frecuencia.

En el caso de ZigBee, cada nodo de la red puede configurarse para aceptar o rechazar datos que no vayan dirigidos al propio nodo, así como puede configurarse cada nodo para retransmitir datos a través de otros nodos en lugar de permitir la conexión directa, aunque sea factible.

Processing / Wiring

Processing es un entorno de programación inspirado en Java y desarrollado en el MIT. Wiring es un lenguaje que se basa en el entorno de Processing y ofrece la posibilidad de programar microcontroladores sin necesidad de conocer su lenguaje ensamblador. Wiring combina el entorno de Processing con un subconjunto del lenguaje C para desarrollar aplicaciones que funcionen en un microcontrolador sin necesidad de conocer el funcionamiento interno de éste.

Utilizamos Processing para el desarrollo del código que se ejecutará en el microcontrolador de la placa Arduino que controlará la máquina.



Spring

El diseño y desarrollo de aplicaciones tiende actualmente a modelos MVC en los que se utiliza una base de datos tradicional pero la vista se implementa como una interfaz web. Java y su tecnología de servlets, junto con el servidor web Tomcat, es una de las plataformas más populares y robustas para implementar este tipo de interfaces, y debido a la popularidad de este tipo de soluciones se han desarrollado diversos frameworks que facilitan al programador la tarea de unir la interfaz web con la lógica de la apliación y la base de datos.

Spring es uno de los frameworks de aplicación más populares para este uso; en concreto, Spring es un framework de apliaciones Java/J2EE. Permite el desarrollo de aplicaciones MVC y facilita tanto el desarrollo de JSP como la comunicación con la base de datos.

Utilizamos Spring para el desarrollo de la aplicación que va a ejecutarse en el servidor, ya que nos facilita una capa de abstracción con la parte de persistencia por un lado, y con la parte de presentación por el otro, y permitiéndonos desarrollar código independiente de la tecnología utilizada para el almacenamiento de información en la base de datos y de representación de información.

Spring proporciona:

- Una potente gestión de configuración basada en JavaBeans, aplicando los principios de Inversión de Control (IoC). Esto hace que la configuración de aplicaciones sea rápida y sencilla. Ya no es necesario tener singletons ni ficheros de configuración, una aproximación consistente y elegante. Esta factoría de beans puede ser usada en cualquier entorno, desde applets hasta contenedores J2EE. Estas definiciones de beans se realizan en lo que se llama el contexto de aplicación.

- Una capa genérica de abstracción para la gestión de transacciones, permitiendo gestores de transacción “enchufables” (pluggables), y haciendo sencilla la demarcación de transacciones sin tratarlas a bajo nivel. Se incluyen estrategias

genéricas para JTA y un único JDBC DataSource. En contraste con el JTA simple o EJB CMT, el soporte de transacciones de Spring no está atado a entornos J2EE.

- Una capa de abstracción JDBC que ofrece una significativa jerarquía de excepciones (evitando la necesidad de obtener de SQLException los códigos que cada gestor de base de datos asigna a los errores), simplifica el manejo de errores, y reduce considerablemente la cantidad de código necesario.

- Integración con Hibernate en términos de soporte a implementaciones DAO y estrategias con transacciones. Especial soporte a Hibernate añadiendo convenientes características de IoC, y solucionando muchos de los comunes problemas de integración de Hibernate. Todo ello cumpliendo con las transacciones genéricas de Spring y la jerarquía de excepciones DAO.

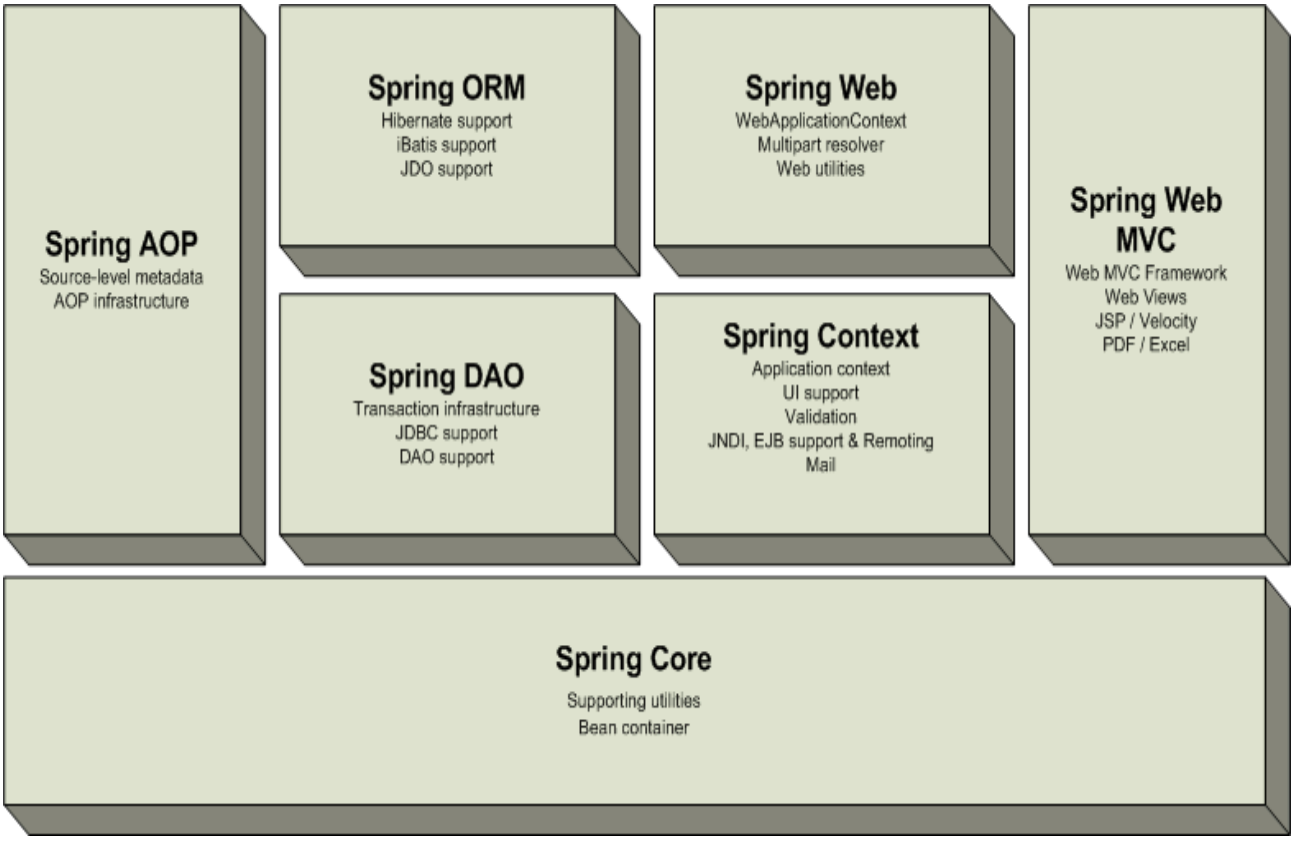
- Funcionalidad AOP, totalmente integrada en la gestión de configuración de Spring. Se puede aplicar AOP a cualquier objeto gestionado por Spring, añadiendo aspectos como gestión de transacciones declarativa. Con Spring se puede tener gestión de transacciones declarativa sin EJB, incluso sin JTA, si se utiliza una única base de datos en un contenedor web sin soporte JTA.

- Un framework MVC (Model-View-Controller), construido sobre el núcleo de Spring. Este framework es altamente configurable vía interfaces y permite el uso de múltiples tecnologías para la capa vista como pueden ser JSP, Velocity, Tiles, iText o POI. De cualquier manera una capa modelo realizada con Spring puede ser fácilmente utilizada con una capa web basada en cualquier otro framework MVC, como Struts, WebWork o Tapestry.

Toda esta funcionalidad puede usarse en cualquier servidor J2EE, y la mayoría de ella ni siquiera requiere su uso. El objetivo central de Spring es permitir que objetos de negocio y de acceso a datos sean reusables, no atados a servicios J2EE específicos. Estos objetos

pueden ser reutilizados tanto en entornos J2EE (web o EJB), aplicaciones standalone, entornos de pruebas,... sin ningún problema.

La arquitectura en capas de Spring ofrece gran cantidad de flexibilidad. Toda la funcionalidad está construida sobre los niveles inferiores. Por ejemplo se puede utilizar la gestión de configuración basada en JavaBeans sin utilizar el framework MVC o el soporte AOP.



Ventajas de Spring:

- Spring provee muchas funcionalidades paquetizadas en diferentes librerías.
- Hace más fácil y comprensivos los desarrollos en J2EE.
- Promueve la aplicación de la buenas prácticas de programación y uso de patrones.

- Aprovecha los desarrollos de librerías opensources ya existentes y permite la integración sencilla de librerías externas.

- Elimina la proliferación de clases Singletone, que dificultan las pruebas y la orientación a objetos.

- Facilita el uso de los ficheros de configuración y elimina la necesidad de construir diferentes formatos de ficheros de configuración.

- Provee un framework para acceso a datos (JDBC, Hibernate, Ibatis, JTA, ect).

- Está diseñado para que las aplicaciones tengan la menor dependencia del framework. La mayoría de los objetos no tienen dependencia con Spring.

- Facilita la creación e integración con servicios Web.

Acegi Security System for Spring

Acegi Security proporciona servicios de seguridad dentro de Spring Framework. Aunque no forma parte directa de Spring está íntimamente ligado con éste.

Proporciona un sistema de autenticación a través del cual los usuarios pueden autenticarse y acceder a múltiples aplicaciones a través de un único punto de entrada. Para ello utiliza el servicio de autenticación CAS (Central Authentication Service) desarrollado por la Universidad de Yale, con el que una aplicación utilizando Acegi Security puede participar en un entorno single sign on a nivel de toda la empresa. Ya no es necesario que cada aplicación tenga su propia base de datos de autenticación, ni tampoco existe la restricción de que sólo se pueda utilizar dentro del mismo servidor de aplicaciones. Otras características avanzadas que proporciona son soporte para proxy y refresco forzado de logins. Sus características más relevantes son:

- Integración completa en Spring: Utiliza los mecanismos de configuración de Spring

- Seguridad a nivel de instancia de objetos del dominio: En muchas aplicaciones es deseable definir listas de control de acceso (Access Control Lists o ACLs) para instancias de objetos del dominio individuales. Proporciona un completo paquete ACL con características que incluyen máscaras de bits, herencia de permisos, un repositorio utilizando JDBC, caché y un diseño pluggable utilizando interfaces.

- Configuración no intrusiva: La totalidad del sistema de seguridad puede funcionar en una aplicación web utilizando los filtros que proporciona. No hay necesidad de hacer cambios especiales o añadir librarías al contenedor de Servlets o EJB.

- Integración opcional con los contenedores: Las características de autenticación y autorización que proporcionan los contenedores Servlet o EJB pueden ser usadas utilizando los adaptadores que se incluyen. Actualmente existe soporte para los principales contenedores: Catalina (Tomcat), Jetty, JBoss y Resin

- Mantiene los objetos libres de código de seguridad: Muchas aplicaciones necesitan proteger datos a nivel de objeto basándose en cualquier combinación de parámetros (usuario, hora del día, autoridad del usuario, método que es llamado, parámetros del método invocado,...). Acegi da esta flexibilidad sin necesidad de añadir código a los objetos de negocio.

- Protección de peticiones HTTP: Además de proteger los objetos, el proyecto también permite proteger las peticiones HTTP. Ya no es necesario depender de restricciones de seguridad definidas en el fichero web.xml. Lo mejor de todo es que las peticiones HTTP pueden ser protegidas por una serie de expresiones regulares o expresiones de paths como las utilizadas por Ant, así como autenticación, autorización y gestores de reemplazo de credenciales para la ejecución como otro usuario, todo ello totalmente pluggable.

- Seguridad del canal: El sistema de seguridad puede redirigir automáticamente las peticiones a un canal de transmisión adecuado. Comúnmente esto se aplica para asegurar que las páginas seguras estarán sólo disponibles sobre HTTPS, y las páginas públicas sobre HTTP, aunque es suficientemente flexible para soportar cualquier tipo de requisitos de "canal". También soporta combinaciones de puertos no usuales y gestores de decisión de transporte pluggables.

- Soporta autenticación HTTP BASIC: Esta autenticación es adecuada para aquellas aplicaciones que prefieren una simple ventana de login del navegador en lugar de un formulario de login. Acegi Security puede procesar directamente peticiones de autenticación HTTP BASIC siguiendo el RFC 1945.

- Librería de etiquetas: Proporciona una librería de etiquetas que puede ser utilizada en JSPs para garantizar que contenido protegido como enlaces y mensajes son únicamente mostrados a usuarios que poseen los permisos adecuados.

- Configuración mediante el contexto de aplicación de Spring o basada en atributos: Permite seleccionar el método utilizado para la configuración de seguridad del entorno, tanto a través del contexto de aplicación de Spring o utilizando atributos en el código fuente utilizando Jakarta Commons Attributes.

- Distintos métodos de almacenamiento de la información de autenticación: Acegi incluye la posibilidad de obtener los usuarios y permisos utilizando ficheros XML, fuentes de datos JDBC o implementando un interfaz DAO para obtener la información de cualquier otro lugar.

- Soporte para eventos: Utilizando los servicios para eventos que ofrece Spring, se pueden configurar receptores propios para eventos como login, contraseña incorrecta y cuenta deshabilitada. Esto permite la implementación de sistemas de auditoría y bloqueo de cuentas, totalmente desacoplados del código de Acegi Security.

- Caché: Utilizando EHCache o otra implementación propia se puede hacer caché de la información de autenticación, evitando que la base de datos o cualquier otro tipo de fuente de información no sea consultado repetidamente.

Interfaz web mediante JSP

La vista de la aplicación que se ejecuta en el servidor se implementa a través de JSP. Esto permite que pueda accederse a la apliación desde cualquier ordenador que disponga de un navegador sin necesidad de instalación, lo que tiene importantes ventajas como la posibilidad de actualizar la versión de la aplicación sin obligar a cada usuario a actualizar, lo que simplifica el soporte técnico a los usuarios y la corrección de errores en la aplicación.

Si bien la tecnología de JSP no es parte integrante del framework Spring, sí es cierto que Spring está orientado al desarrollo de aplicaciones web y hace un uso intensivo de esta tecnología, integrándola dentro de la estructura de un proyecto Spring.

Mas adelante se muestra la interfaz web, separada por módulos.

Conexión entre Arduino y la aplicación

El módulo de comunicación con el hardware debe conectarse a la placa Arduino. Esto se consigue mediante una biblioteca Java, RXTX, que permite la comunicación con el puerto USB de cualquier aplicación, con mínima dependencia del sistema operativo exceptuando el nombre del dispositivo y algunos detalles de funcionamiento. Gracias a esta biblioteca nuestra aplicación puede ser multiplataforma con mínimos ajustes en la forma de comunicación, siempre que el sistema operativo esté soportado por RXTX.

Hibernate

Hibernate es una biblioteca que automatiza las conexiones con la base de datos. Permite también abstraer la sintaxis específica de la base de datos elegida y por tanto cambiar la base de datos si en cierto momento se precisara.

Técnicamente, Hibernate se define como "object-relational mapping", es decir, permite asociar objetos de un lenguaje de programación orientado a objetos con tablas de una base de datos. Puede decirse que Hibernate permite la persistencia de objetos, en lugar de tener que convertir la información que contienen estos objetos, uno por uno, a un formato que pueda ser guardado en una base de datos. Es la solución ORM (Object-Relational Mapping) más popular en el mundo Java.

Hibernate permite desarrollar clases persistentes a partir de clases comunes, incluyendo asociación, herencia, polimorfismo, composición y colecciones de objetos. El lenguaje de consultas de Hibernate HQL (Hibernate Query Language), diseñado como una mínima extensión orientada a objetos de SQL, proporciona un puente elegante entre los mundos objetual y relacional. Hibernate también permite expresar consultas utilizando SQL nativo o consultas basadas en criterios.

Soporta todos los sistemas gestores de bases de datos SQL y se integra de manera elegante y sin restricciones con los más populares servidores de aplicaciones J2EE y contenedores web, y por supuesto también puede utilizarse en aplicaciones standalone. Algunas características clave son:

- Persistencia transparente: Hibernate puede operar proporcionando persistencia de una manera transparente para el desarrollador.

- Modelo de programación natural: Hibernate soporta el paradigma de orientación a objetos de una manera natural: herencia, polimorfismo, composición y el framework de colecciones de Java.

- Soporte para modelos de objetos con una granularidad muy fina: Permite una gran variedad de mapeos para colecciones y objetos dependientes.

- No es necesaria la generación de código ni el procesamiento del bytecode en el proceso de compilación.

- Escalabilidad extrema: Hibernate posee un alto rendimiento, tiene una caché de dos niveles y puede ser usado en un cluster. Permite inicialización perezosa (lazy) de objetos y colecciones.

- Lenguaje de consultas HQL: Este lenguaje proporciona una independencia del SQL de cada base de datos, tanto para el almacenamiento de objetos como para su recuperación.

- Soporte para transacciones de aplicación: Hibernate soporta transacciones largas (aquellas que requieren la interacción con el usuario durante su ejecución) y gestiona la política optimistic locking automáticamente.

- Generación automática de claves primarias: Soporta los diversos tipos de generación de identificadores que proporcionan los sistemas gestores de bases de datos (secuencias, columnas autoincrementales,...) así como generación independiente de la base de datos, incluyendo identificadores asignados por la aplicación o claves compuestas.

Utilizamos Hibernate como una abstracción de la capa de persistencia que nos permite que todo nuestro código sea independiente de la base de datos utilizada, y que además simplifica la gestión de transacciones.

Tomcat / JSP

Tomcat es un servidor web desarrollado por la Apache Foundation que se basa en el servidor Apache HTTPD y que incluye soporte para JSPs.

JSP, o Java Server Pages, es una tecnología basada en Java que permite crear páginas web dinámicas que se generan automáticamente cada vez que un usuario accede a una URL o efectúa una acción en una página. JSP permte que se ejecute cierto código escrito en Java en respuesta a cada una de estas acciones, permitiendo en efecto cambiar al contenido de una página según el usuario que la solicita y según la petición.

JSP incluye una sintaxis específica en las páginas web, que posteriormente el servidor web Tomcat interpretará ejecutando el código definido usando esta sintaxis especial. Por lo demás, las páginas web que presenta Tomcat son estándar y no requieren ninguna modificación posterior.

Utilizamos Tomcat como servidor web para nuestra aplicación, ya que permite tanto la utilización de servlets como de JSP. Sin Tomcat no podríamos desarrollar páginas dinámicas que mostrasen diferente información según la petición del usuario basándonos en Java/J2EE, sólo podríamos mostrar páginas estáticas con contenido fijo para todos los usuarios.

Maven

Maven es una herramienta de gestión de información de proyectos. Maven está basado en el concepto de un modelo de objetos del proyecto POM (Project Object Model) en el que todos los productos (artifacts) generados por Maven son el resultado de consultar un modelo de proyecto bien definido. Compilaciones, documentación, métricas sobre el código fuente y un innumerable número de informes son todos controlados por el POM.

Maven tiene muchos objetivos, pero resumiendo Maven nos ha proporcionando una estructura de proyecto bien definida. Maven aligera en gran cantidad lo que la mayoría de desarrolladores consideran trabajo pesado y aburrido y les permite proseguir con la tarea. Esto es esencial en proyectos open source donde no hay mucha gente dedicada a la tarea de documentar y propagar la información crítica sobre el proyecto que es necesaria para atraer potenciales nuevos desarrolladores y clientes.

La ambición de Maven es hacer que el desarrollo interno del proyecto sea altamente manejable con la esperanza de proporcionar más tiempo para el desarrollo entre proyectos. Se puede llamar polinización entre proyectos o compartir el conocimiento sobre el desarrollo del proyecto.

Características:

- El modelo de objetos del proyecto POM es la base de cómo Maven trabaja. El desarrollo y gestión del modelo está controlado desde el modelo del proyecto.

- Un único conjunto de métodos son utilizados para todos los proyectos que se gestionan. Ya no hay necesidad de estar al tanto de innumerables sistemas de compilación. Cuando las mejoras se hacen en Maven todos los usuarios se benefician.

- Integración con Gump, una herramienta usada en el proyecto Jakarta para ayudar a los proyectos a mantener compatibilidad con versiones anteriores.

- Publicación del sitio web basado en el POM. Una vez el POM es exacto los desarrolladores pueden publicar fácilmente el contenido del proyecto, incluyendo la documentación personalizada más el amplio conjunto de documentación generada por Maven a partir del código fuente.

- Publicación de distribuciones basada en el POM.

- Maven alenta el uso de un repositorio central de librerías, utilizando un mecanismo que permite descargar automáticamente aquellas necesarias en el proyecto, lo que permite a los usuarios de Maven reutilizar librerías entre proyectos y facilita la comunicación entre proyectos para asegurar que la compatibilidad entre distintas versiones es correctamente tratada.

- Guías para la correcta disposición de los directorios. Maven contiene documentación sobre como disponer los directorios de forma que una vez es aprendida se puede ver fácilmente cualquier otro proyecto que use Maven.

JasperReports

Muchas aplicaciones que utilizan bases de datos requieren la representación de los datos almacenados. JasperReports es una biblioteca para Java que facilita la tarea de representar datos exteraídos de una base de datos y exportarlos a diferentes formatos usables por otras aplicaciones como DOC o XLS, todo esto sin necesidad de crear nuestro propio código para la extracción de este conjunto de datos.

JasperReports se utiliza para generar los informes de sesiones en formato Excel.

jQuery

Para algunas representaciones gráficas, especialmente aquellas con las que el usuario puede interactuar, usaremos jQuery. Esta herramienta es una biblioteca para JavaScript que hace más fácil al programador desarrollar páginas con elementos que permiten al usuario introducir información o visualizarla dinámicamente, algo que el HTML estándar no gestiona con facilidad.

En este caso utilizamos una aplicación jQuery llamada Flot cuyo propósito es dibujar y permitir interacción con gráficos sencillos dentro de una página web escrita en HTML o XHTML. Flot utiliza JavaScript para dibujar el gráfico y recoger las pulsaciones del ratón que realiza el usuario, e inserta este gráfico dentro del documento HTML mediante el elemento "canvas" que proporciona este lenguaje de marcado. Usando el propio JavaScript se puede dibujar dentro del "canvas" usando su propia API.

Utilizamos jQuery para representar el plan de trabajo de un usuario y permitirle modificar los valores fijados pulsando con el ratón sobre las gráficas en lugar de tener que modificar a mano los valores numéricos presentados en los correspondientes campos de texto.

Applets Java

Un applet Java es una pequeña aplicación que usualmente se ejecuta usando como contenedor el navegador web del usuario. La ventaja principal de un applet es que puede implementar funcionalidades que una página web normal no aporta, como por ejemplo la representación de gráficos, que es precisamente el uso que se le da en este proyecto.

Tiene además la posibilidad de actuar independientemente de la página; cuando una página web normal acaba de cargarse su contenido no puede actualizarse si no es usando tecnologías diferentes de HTML o XHTML, como por ejemplo JavaScript, o bien obligando a que la página se refresque automáticamente cada cierto intervalo de tiempo. Nuestro applet puede redibujar su contenido cada vez que encuentre nuevos datos para representar sin necesidad de recargar la página.

Para la representación gráfica utilizamos una sencilla biblioteca de gráficos que permite representar puntos en dos dimensiones, llamada OpenChart2. Sólo necesita recibir un array de puntos, y los representa en un eje de coordenadas.

Conclusiones y ampliaciones futuras

El hecho que nos llevó a iniciar este proyecto fue la aparición de dispositivos hardware sencillos y baratos destinados al control de elementos físicos, especialmente en el campo de la domótica, y la carencia de soluciones integrales que aunaran hardware, comunicación entre dispositivos y software de gestión y control del hardware antes mencionado. Nuestro proyecto intenta rellenar este hueco teniendo siempre en mente la extensibilidad a todo tipo de necesidades de control.

En concreto, este proyecto se ha centrado en gestionar las máquinas de un gimnasio, mercado en el que las tecnologías de control aún no están muy implicadas, posiblemente por su alto coste hasta ahora. Las máquinas de un gimnasio, si bien disponen de sistemas de control más o menos sofisticados, no tienen en general forma de comunicarse con una aplicación central que pueda controlar el entrenamiento de cada usuario y modificar sus planes de entrenamiento. Es aquí donde este proyecto intenta cubrir una necesidad ya existente, y hacerlo a un coste asequible para la casi totalidad de negocios.

Desde un punto de vista más general, aún no es habitual fuera de las grandes plantas robotizadas el control de máquinas y otros elementos físicos de forma remota, y es aún menos común el control inalámbrico de bajo coste unido a una aplicación de gestión que permita programar acciones en esos dispositivos o recoger información de ellos y almacenarla.

En lo relativo al diseño y construcción del módulo de hardware, podemos afirmar que el desarrollo con Arduino, utilizando su IDE y aprovechándonos de los módulos ya existentes para esta placa como el módulo XBee o incluso la conexión mediante USB, ha resultado menos complejo de lo que habría sido si hubiéramos tenido que programar varios microcontroladores y diseñar la conexión con todos los periféricos que estamos utilizando.

Hemos observado además que el uso de Spring con Hibernate para la aplicación web no

sólo nos ha facilitado la tarea de desarrollar en Java una aplicación con persistencia en una base de datos y con interfaz web, sino que además nos ha permitido añadir una capa de abstracción respecto de estas dos tecnologías, de modo que en un futuro podríamos potencialmente modificar tanto la capa de persistencia -por ejemplo cambiando de base de datos- como la capa de presentación -por ejemplo usando una presentación convencional en Swing en lugar de vía web- sin hacer cambios en la parte de lógica de negocio.

Las futuras ampliaciones podrían ir dirigidas en dos direcciones fundamentalmente: la primera es la gestión de dispositivos más complejos que requiriesen de formas de control y programación más complicadas y con una mayor necesidad de interacción entre dispositivos. Un buen ejemplo sería la simulación de una planta industrial en la que existieran necesidades de comunicación entre todas las máquinas para que una de ellas se arrancase en caso de fallo de alguna otra, así como una capa de seguridad que impidiese a intrusos leer la información transmitida por la red.

El otro campo que se abre para posibles ampliaciones es el de la inteligencia artificial. Podría adapatarse la aplicación para tomar decisiones ante señales inesperadas que enviasen los dispositivos, teniendo en cuenta un conjunto de objetivos. Por ejemplo, podría diseñarse una casa domótica que reaccionara ante fenómenos imprevistos como una lluvia torrencial o la intrusión de un individuo no autorizado.

Plan de trabajo

Para el desarrollo de este proyecto hemos dividido el tiempo en siete grandes segmentos: toma de requisitos, diseño, implementación, pruebas, arreglo de defectos, integración, segunda fase de pruebas y segunda fase de arreglo de defectos. Debido a que las primeras fases de pruebas y arreglo de defectos se realizan de forma independiente con cada módulo, aparecen aquí insertadas en la fase de implementación, para no tener que describir cada una de estas fases por cada módulo que se ha implementado.

Esta división en bloques intenta corresponderse con un modelo iterativo de desarrollo de software; pero debido a que este proyecto se diferencia de otros proyectos típicos en que los requisitos se definen desde el principio y no sufren cambios importantes a lo largo del proceso, decidimos eliminar fases de verificación y cambio o nueva recogida de requisitos ya que asumimos que no iban a ser necesarias.

Fase de toma de requisitos

Del 10 al 31 de octubre de 2008 hicimos la recogida de requisitos mediante dos reuniones con el director del proyecto y la confección de tres documentos relacionados: reparto tentativo del trabajo, descripción preliminar de la parte software y descripción preliminar de la parte hardware. En lo relativo al reparto de las tareas, decidimos que Javier Murillo se encargaría del módulo de hardware mientras que Helena Lorenzo y Alberto Velázquez desarrollarían la aplicación que se ejecuta en el servidor.

Fase de diseño

Del 1 al 24 de noviembre de 2008 diseñamos por un lado los detalles de la aplicación web, y por otro lado el módulo de hardware y la comunicación entre hardware y software.

Para la aplicación web diseñamos el reparto de las funcionalidades en módulos, así como el modelo de la base de datos. Para la parte hardware diseñamos la máquina de estados que controla la secuencia de acciones que realiza el hardware. Fijamos también en común el sencillo protocolo de comunicación entre el módulo hardware y la aplicación.

Con un conjunto de tareas más claro detallamos mejor el reparto de trabajo: Javier Murillo se encarga del módulo hardware, codificación del microcontrolador, máquina de estados e integración de los diferentes dispositivos hardware que lo componen. Helena Lorenzo se encarga de la configuración de Spring e Hibernate para la aplicación, diseño web y del módulo de usuarios, informes y plan de trabajo. Alberto Velázquez se encarga del módulo de máquinas, la parte de estadísticas con el Applet Java que muestra el registro del ejercicio, los gráficos interactivos del módulo de plan de trabajo y la confección de esta memoria con contribuciones de Helena Lorenzo y Javier Murillo.

Fase de implementación, prueba y depuración de módulos por separado

Del 24 de noviembre de 2008 al 5 de abril de 2009 se codificó e integró el código correspondiente a ambos módulos del proyecto.

Los dos primeros módulos de la aplicación, usuarios y máquinas, se desarrollaron del 24 de noviembre de 2008 al 18 de enero de 2009 y fueron probados y depurados hasta el 21 de diciembre. La primera parte del código hardware, la conexión con el PC a través de USB, se desarrolló y probó en las mismas fechas.

Los dos módulos siguientes, gráfica de ejercicios y módulo de informes, se desarrollaron y probaron hasta el 18 de marzo de 2009, coincidiendo con el desarrollo en la parte hardware del control de dispositivos externos: botones, teclado numérico y display.

Del 18 de marzo al 5 de abril se desarrolló el módulo de informes y los gráficos interactivos de la aplicación, y la implementación de la máquina de estados en el módulo hardware.

Fase de integración

Del 5 de abril al 28 de mayo se integrraron ambos módulos, lo que requirió el desarrollo y prueba de código destinado a la interconexión de ambas partes, hardware y aplicación. Se desarrolló además una primera versión de la documentación.

Segunda fase de pruebas y depuración

Del 1 de junio al día previo a la entrega se probará la aplicación completa en busca de defectos, y se desarrollará la versión final de la documentación.

Integración

El proyecto consta de dos grandes bloques: el código que se ejecuta en el microcontrolador de Arduino y la aplicación que corre en el servidor.

El código Arduino ha sido desarrollado enteramente por una única persona directamente usando placas de prueba, de tal forma que no hay que integrar partes diferentes.

La aplicación del servidor es más compleja; está compuesta por diferentes tecnologías (Spring con JSP, Hibernate, Applets Java, JavaScript y jQuery), y está dividida en módulos diferenciados (usuarios, máquinas, sesiones de trabajo, entrenamientos, informes, estadísticas) que han sido desarrollados por dos miembros diferentes del equipo tal y como se detalla en el plan de trabajo, por lo que la integración de esta parte ha sido más compleja.

El código correspondiente al Applet Java se ha desarrollado de forma independiente al resto de la aplicación ya que sólo era necesario conocer la estructura de la base de datos que se fijó en la etapa de diseño de la aplicación; la misma situación se da con los informes generados con JasperReports. Otro tanto ocurre con el código JavaScript con jQuery que se usó para crear gráficos con los que se pudiera interactuar, que se diseñó como un componente “drop-in” para el que sólo era necesario conocer el diseño HTML de la página que iba a mostrar los datos numéricos a representar. Cada uno de estos fragmentos de códifo se desarrolló por un sólo miembro, y su integración en el código de la aplicación fue casi inmediata.

El código desarrollado sobre Spring y dividido en módulos sí requirió un trabajo de integración más cuidadoso, trabajo en el que hubo que invertir menos esfuerzo debido al hecho de que desde un principio establecimos el uso de un sistema de control de versiones denominado Subversion. Este programa, que se puede integrar como un módulo en el entorno de desarrollo Eclipse o se puede usar como un programa autónomo, permite que varios programadores trabajen sobre las mismas partes del código a la vez, algo que tuvimos

que hacer a menudo. El código y todos los cambios realizados en él se almacenan en un servidor en Internet llamado “repositorio”, y cuando un desarrollador quiere comenzar a trabajar debe primero comprobar que tiene los últimos cambios realizados al código. Todos los desarrolladores tienen una copia local del código sobre la que hacen cambios, y cuando acaban los suben al repositorio. Si el sistema de gestión de versiones detecta que se ha modificado a la vez un mismo archivo, presenta al desarrollador los cambios en conflicto y le permite decidir qué hacer.

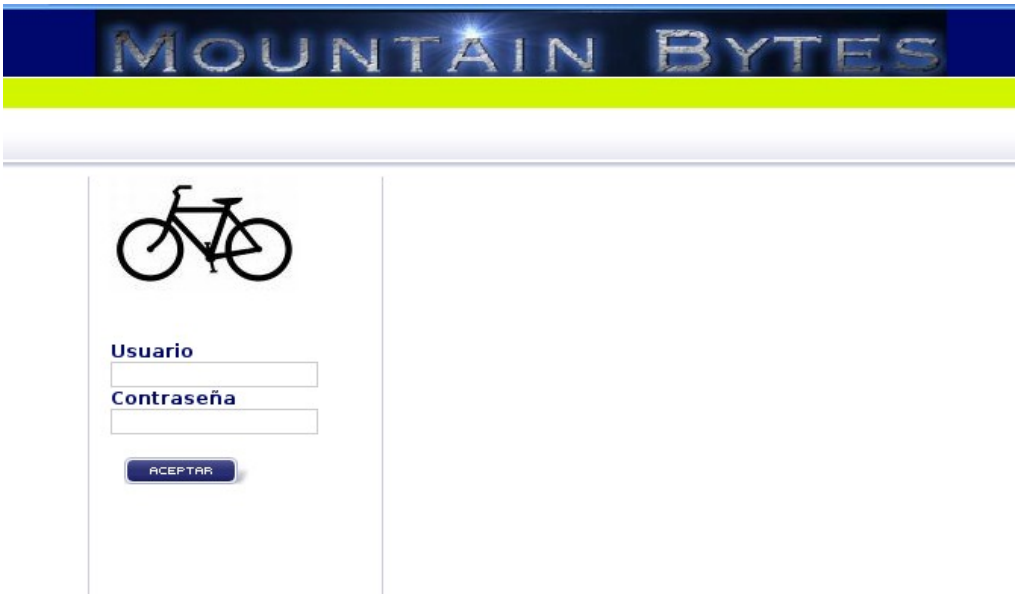
Usando este sistema de control de versiones pudimos garantizar que el código funcionaba adecuadamente después de cada cambio, lo que permitió no tener que comprobar al final de la etapa de desarrollo si todos los módulos funcionaban bien al integrarlos.

El mayor esfuerzo de integración fue la comunicación entre el código de Arduino y la aplicación del servidor. En concreto, el módulo de gestión de sesiones debe ser capaz de comunicarse mediante el puerto USB con Arduino. Para ello se utiliza una biblioteca de conexión de Java, RXTX, que permite que una aplicación Java use el puerto USB. Además, debido a su naturaleza multiplataforma, permite también que la aplicación funcione en diversos sistemas operativos con mínimos cambios.

Manual de usuario

Interfaz completa de la aplicación

La aplicación que se ejecuta en el servidor dispone de una interfaz web, y está dividida en diferentes módulos que agrupan las diferentes funcionalidades ofrecidas. La primera pantalla que encuentra el usuario es el login:



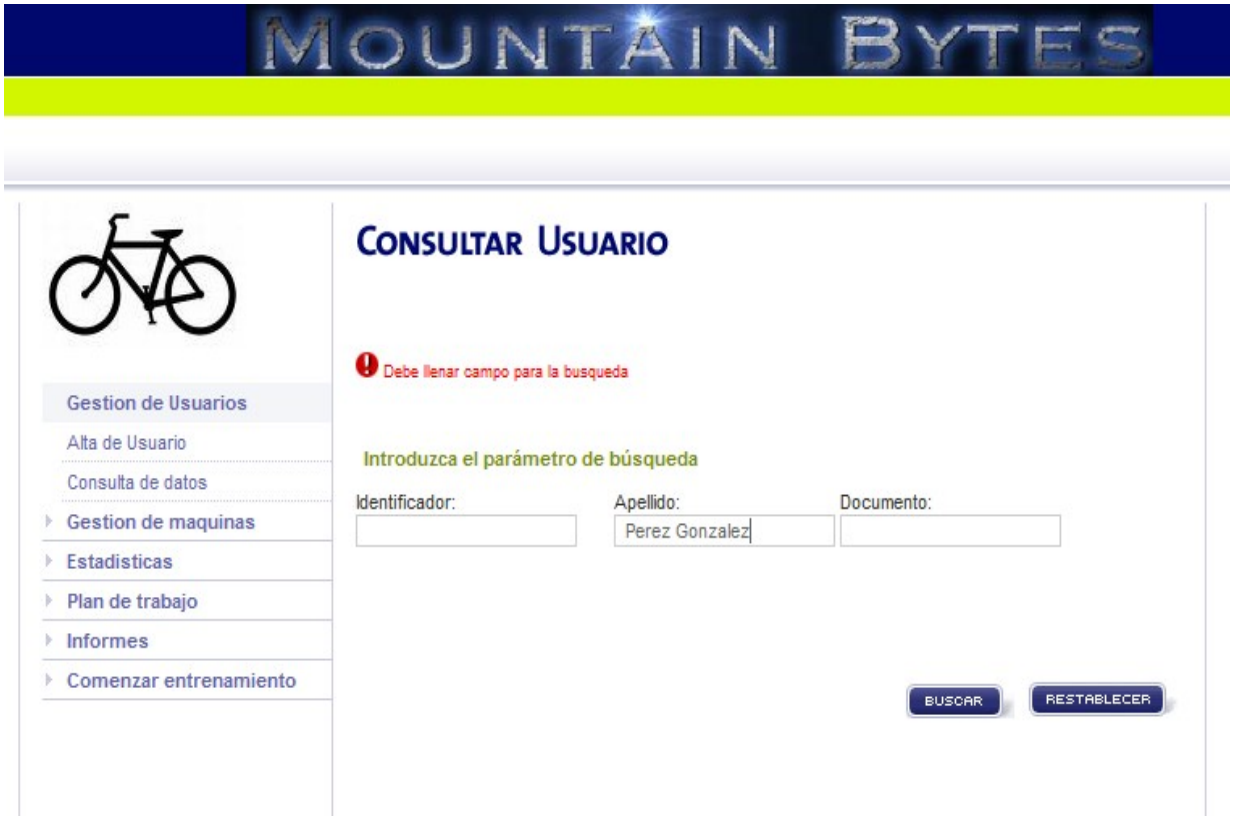
En la pantalla de login, si el usuario no introduce un nombre y contraseña válidos recibe un mensaje de error indicándole el problema.

Cuando el usuario accede a la aplicación, la primera opción que se le ofrece es modificar su perfil:

Si el usuario desea cambiar sus datos sólo deberá introducir la nueva información y pulsar en “modificar”. Los usuarios administradores pueden también consultar modificar los datos de otros usuarios seleccionando la opción “consulta de datos” que sólo se muestra si el perfil del usuario actual corresponde al de administrador.

En caso de ser administrador podemos hacer una búsqueda de usuarios bien por el identificador, o por el apellido o por su número de documento. Esta pantalla de búsqueda no la tiene un usuario normal, el usuario normal al dar a consulta le aparecen sus datos únicamente, es decir es como si le apareciese el resultado del administrador al realizar una búsqueda.

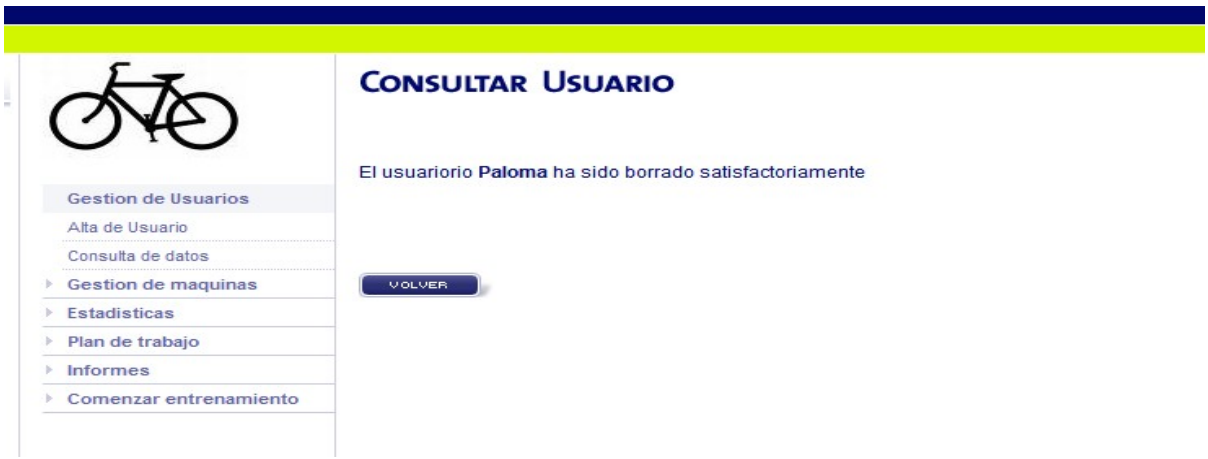
Si no se indicó campo de búsqueda, recibiremos un error:



De lo contrario, veremos los datos correspondientes:

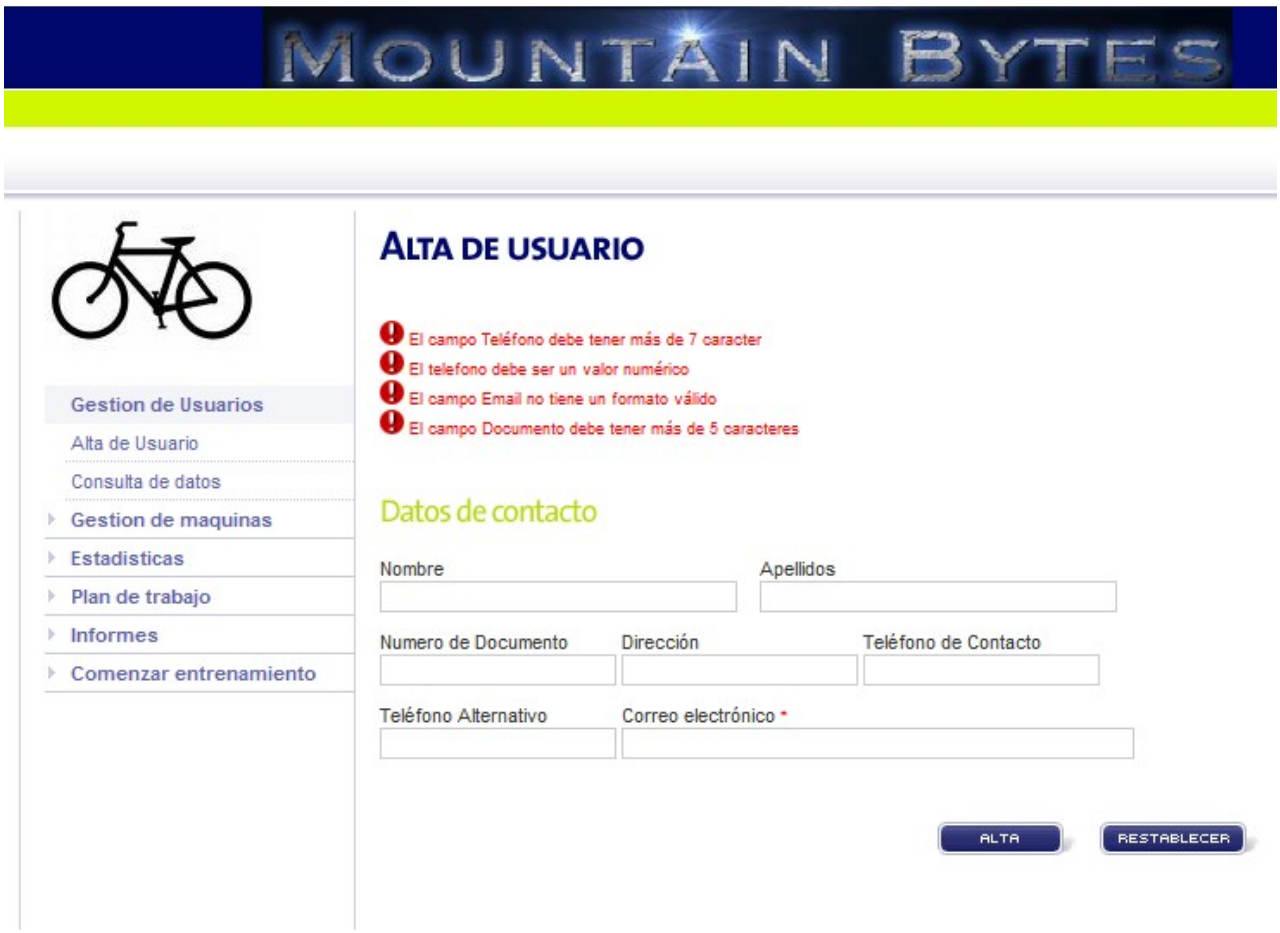
A través de esta pantalla podemos modificar el usuario seleccionado, las reglas de validación de los campos son las mismas que en la creación de los mismos. El botón volver nos devuelve a la pantalla anterior, el restablecer nos pone los datos que estaban inicialmente y el modificar nos modifica los datos del usuario en la base de datos. Un usuario no administrador sólo puede modificar sus datos ya que no tiene pantalla de búsqueda de usuarios y le aparecen sus datos únicamente.

Tras pulsar en el botón de eliminar el usuario es eliminado del sistema, esta opción sólo se le muestra al administrador del sistema. La acción da como resultado:

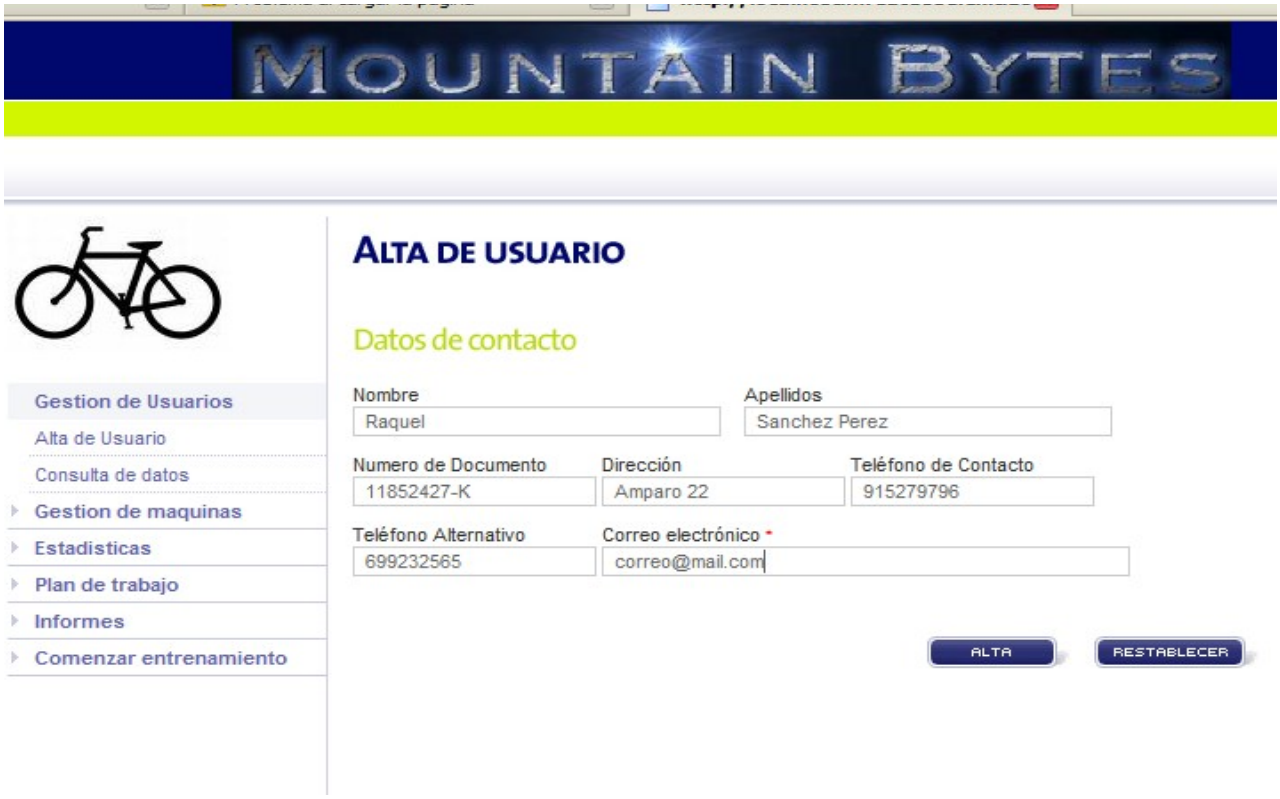


La pantalla de alta de usuario nos permite dar de alta usuarios en el sistema. Mediante el botón restablecer volvemos a poner todos los campos en blanco para empezar de nuevo y con el botón aceptar lo que hacemos es dar al usuario de alta en el sistema tras las validaciones correspondientes. Dichas validaciones son:

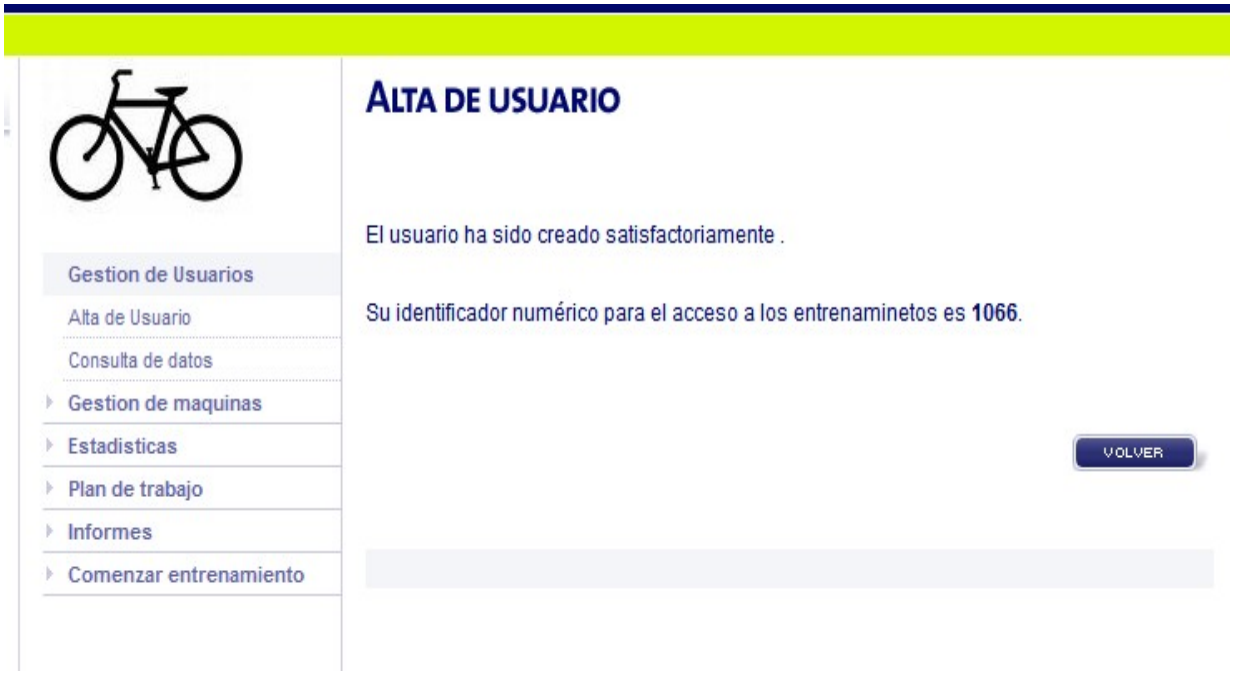
- El nombre del usuario no puede ser vacío.
- El apellido no puede ser vacío.
- El teléfono no puede ser vacío y debe estar compuesto por sólo números y tener una longitud mínima de 7 caracteres.
- El campo teléfono alternativo puede estar en blanco, pero si se rellena debe contener todo números y una longitud mínima de 7 caracteres.
- El email no puede estar vacío y debe tener un formato de email.
- El número de documento es un campo obligatorio y además, se valida que el DNI tiene un formato válido y es un DNI real.



El proceso de alta de usuario además de dar al usuario de alta le pone por defecto como clave de acceso al sistema el DNI del usuario, y le asigna un número aleatorio y no existente en la base de datos como clave de acceso para los entrenamientos en las máquinas.



Ante un alta correcta, recibimos el mensaje correspondiente:



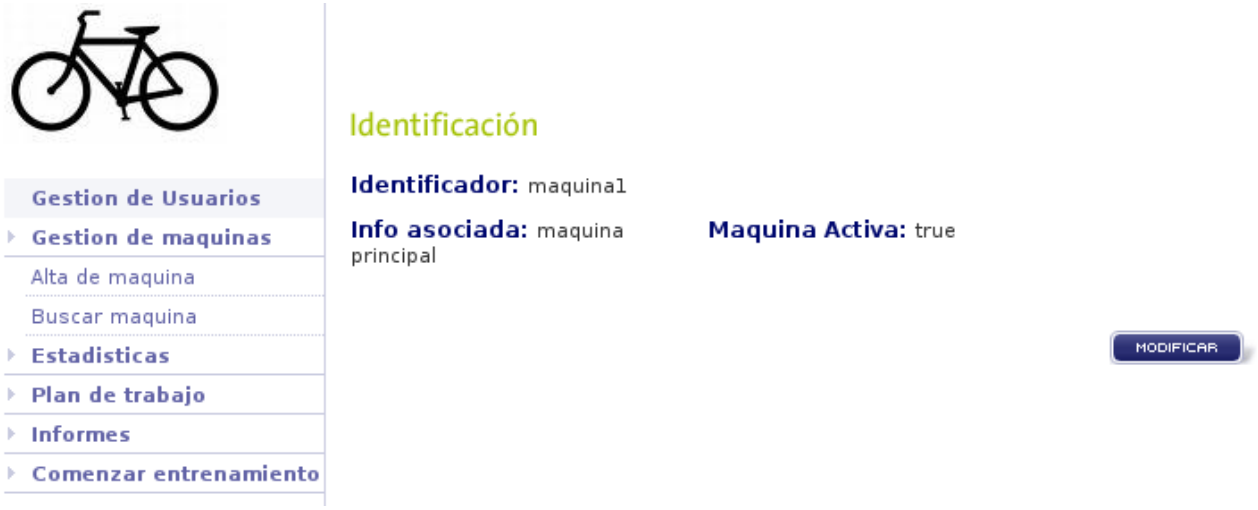
El siguiente módulo, muy parecido al primero, es el de gestión de máquinas. Por defecto, al seleccionar este módulo aparece la pantalla de consulta de máquina:

Esta pantalla permite gestionar diferentes máquinas, buscándolas por nombre. Antes de poder buscar máquinas deberemos crear una, por lo que seleccionamos la opción "alta de máquina":

Aquí introduciremos un nombre que identifique a la máquina en nuestra base de datos, así como información opcional que identifique la máquina o aporte información sobre la misma. Cuando hayamos acabado pulsaremos en "alta"; si nos hemos equivocado al introducir la información y queremos borrar todos los campos, pulsaremos en "restablecer". Si no hubo problemas al dar de alta la máquina, el sistema nos informará de que la máquina ha sido dada de alta satisfactoriamente. Por defecto, las máquinas están activas. Si la

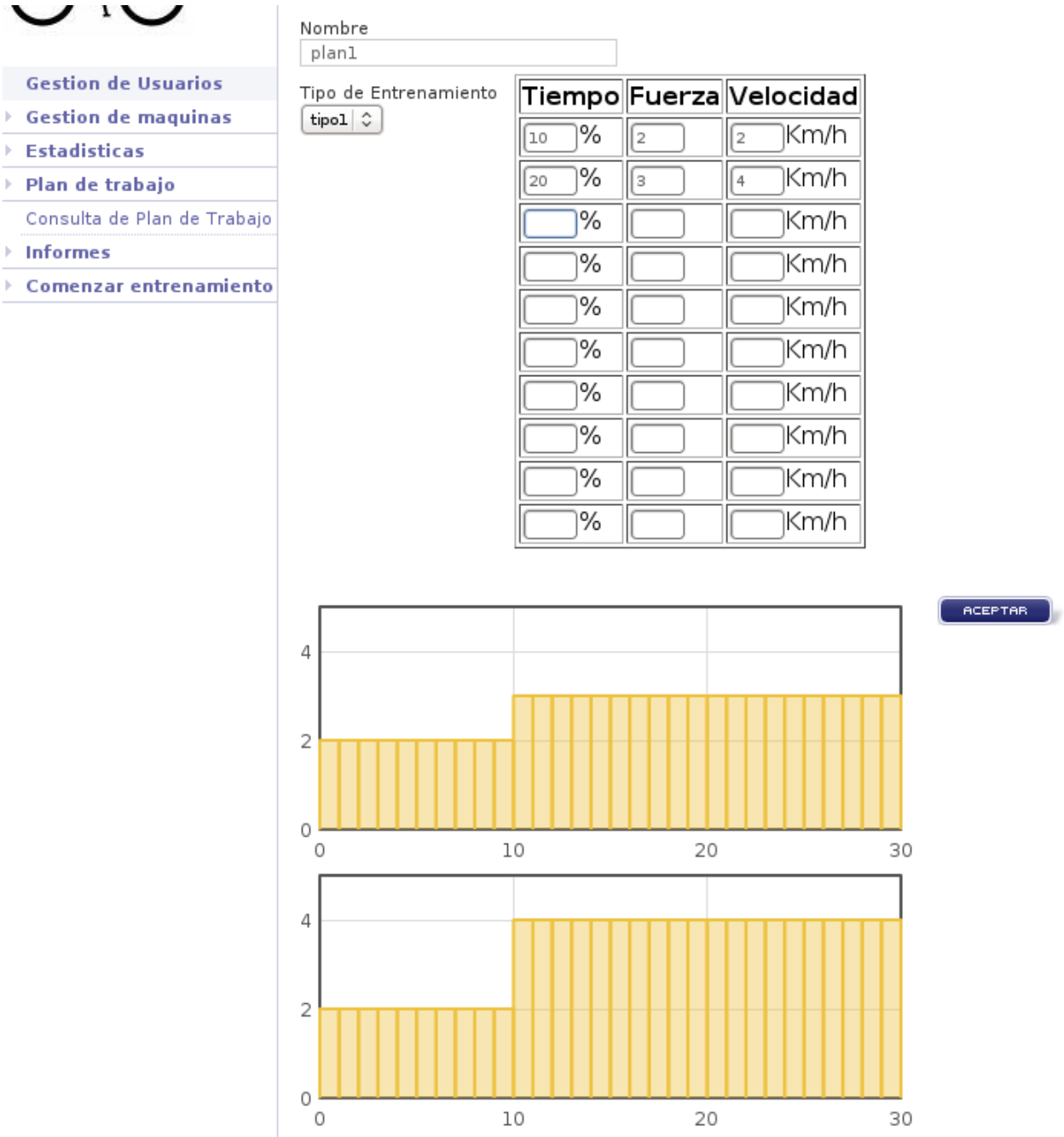
máquina se retira o no está disponible temporalmente, se deberá modificar su estado e indicar que no está activa mediante la opción de modificar máquina.

Ahora ya podemos buscar la máquina que acabamos de crear a través de la opción "buscar máquina". Si introducimos un nombre de máquina inexistente la aplicación nos informará de ello. En caso contrario, nos mostrará la información asociada a la máquina que hemos creado.

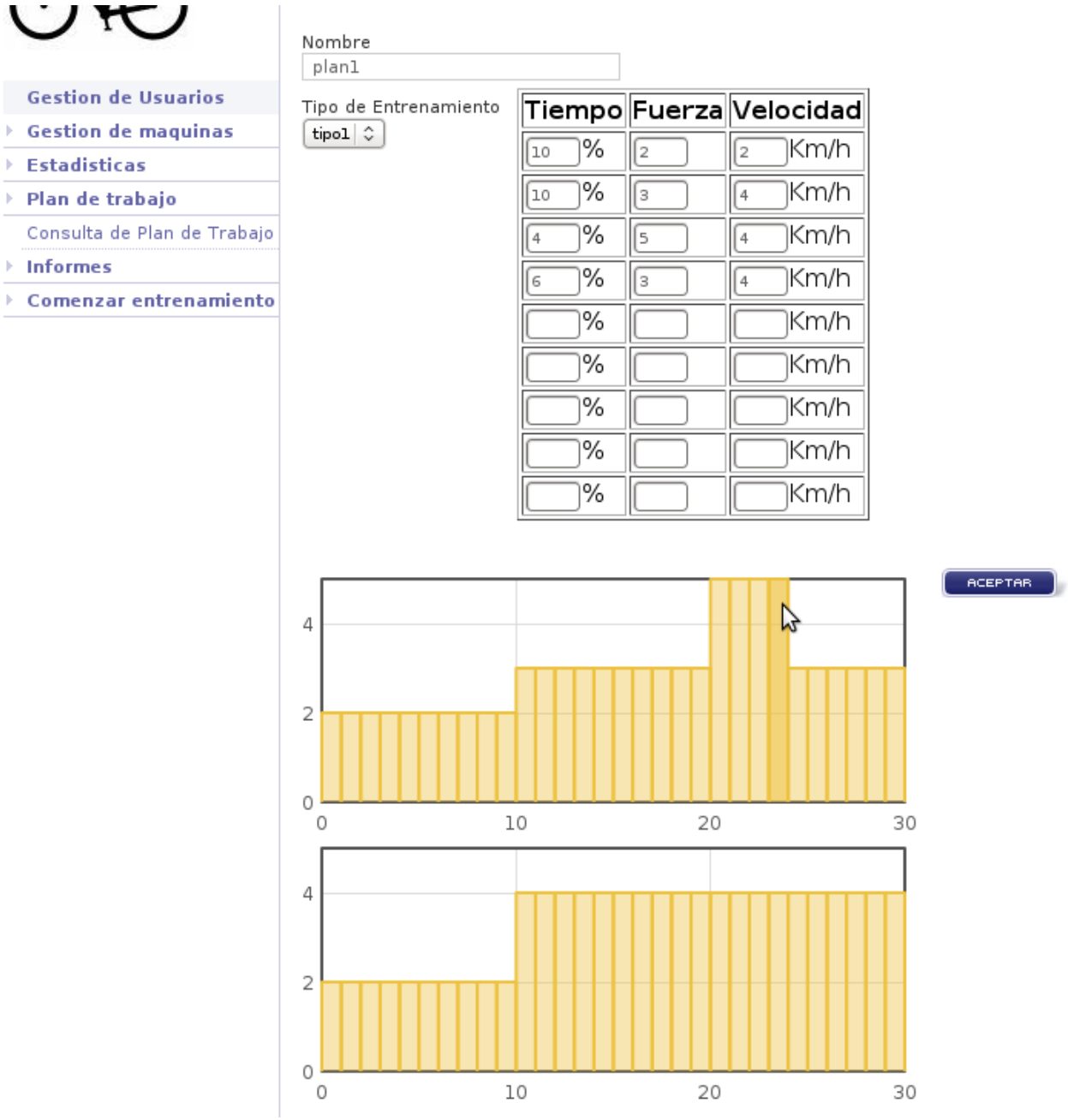


Si deseamos cambiar la información asociada a la máquina sólo tenemos que pulsar el botón "modificar", que nos permitirá alterar los diferentes campos asociados a esta máquina:

El siguiente módulo que veremos es el de plan de trabajo. Este módulo permite la creación y edición por parte del usuario o de un administrador de un plan de trabajo en la máquina. Permite en el caso de que la máquina sea una bicicleta estática definir cuánto tiempo durará una sesión, y permitirá también definir la velocidad y resistencia de la máquina en cada intervalo de tiempo. Estos parámetros pueden ajustarse introduciendo valores numéricos en los campos correspondientes, o bien pulsando directamente en los gráficos que aparecerán en la parte inferior de la pantalla y que permitirán ajustar velocidad y resistencia respectivamente en el intervalo de tiempo definido. Al actuar sobre estos gráficos se modificarán automáticamente los valores numéricos en los campos correspondientes.

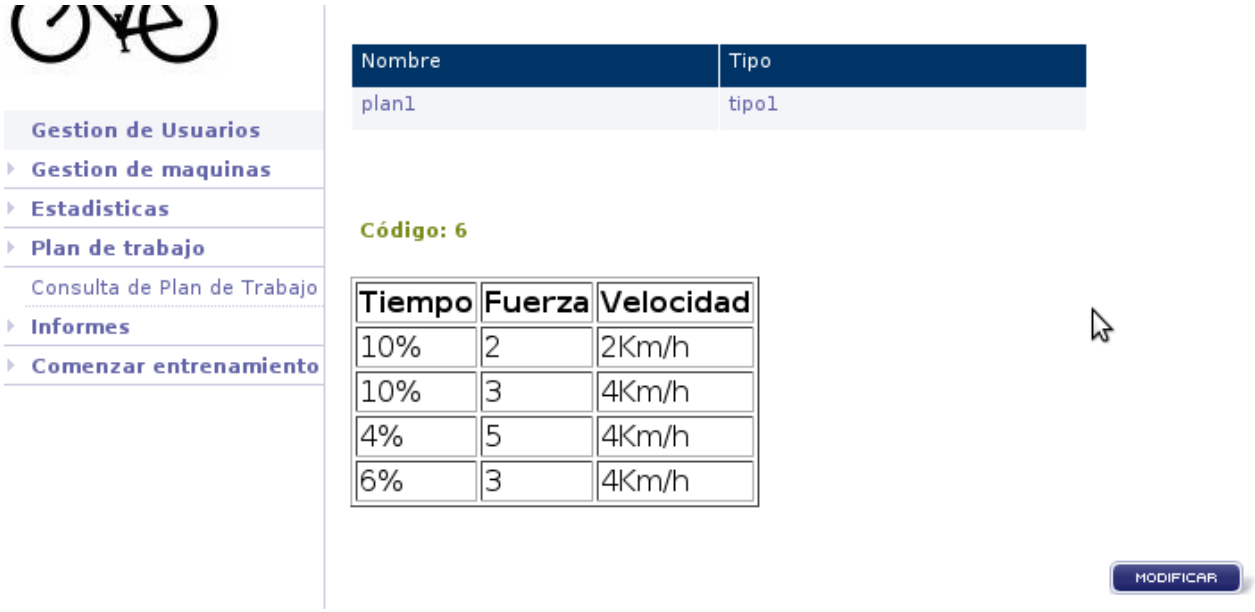


Podemos ver en la siguiente captura el efecto de pinchar sobre los gráficos, que modifica automáticamente los valores de los campos:

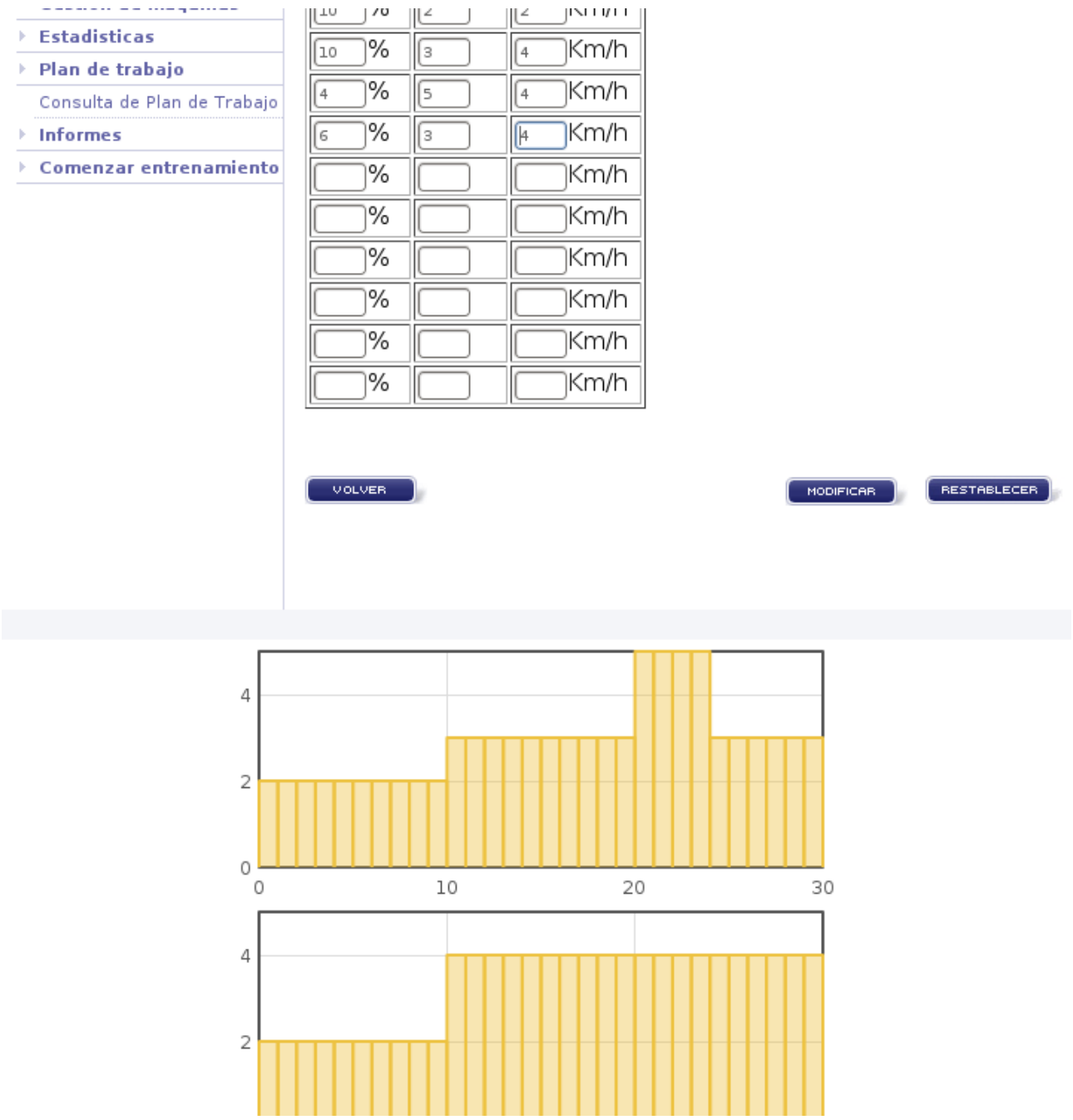


Naturalmente también existe la posibilidad de buscar entre los planes de trabajo ya creados, pantalla que es similar a la búsqueda en el resto de módulos. Si la búsqueda ha tenido éxito se devuelven los planes existentes que coincidieron con los parámetros indicados:

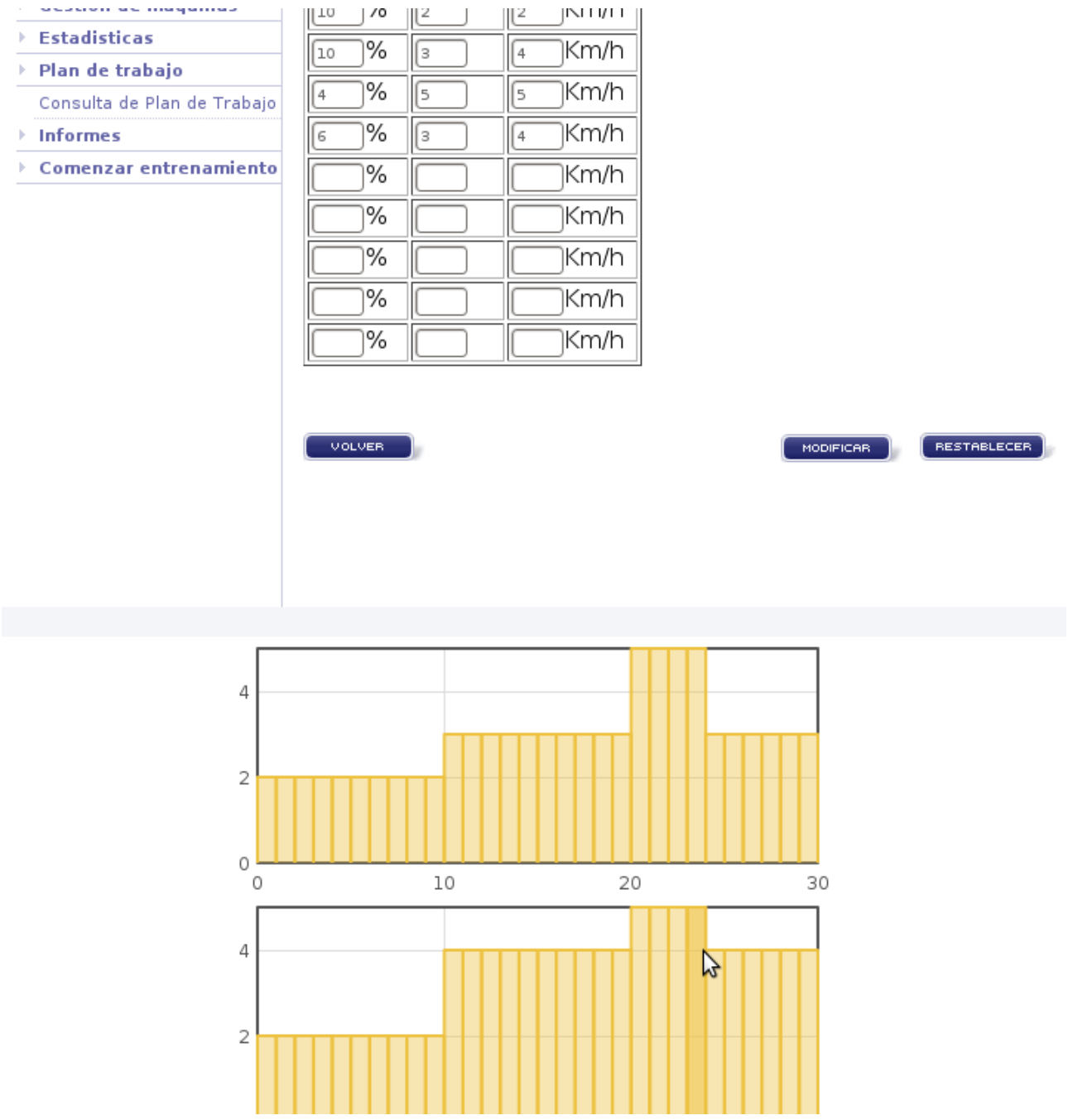
Pulsando sobre el plan deseado podremos acceder a los detalles del plan:



Pulsando en "modificar" accederemos a una pantalla similar a la de creación que nos permitirá cambiar los detalles de este plan de trabajo, bien modificando los valores numéricos o bien actuando sobre las gráficas:



Vemos también el efecto de pinchar directamente sobre la gráfica:

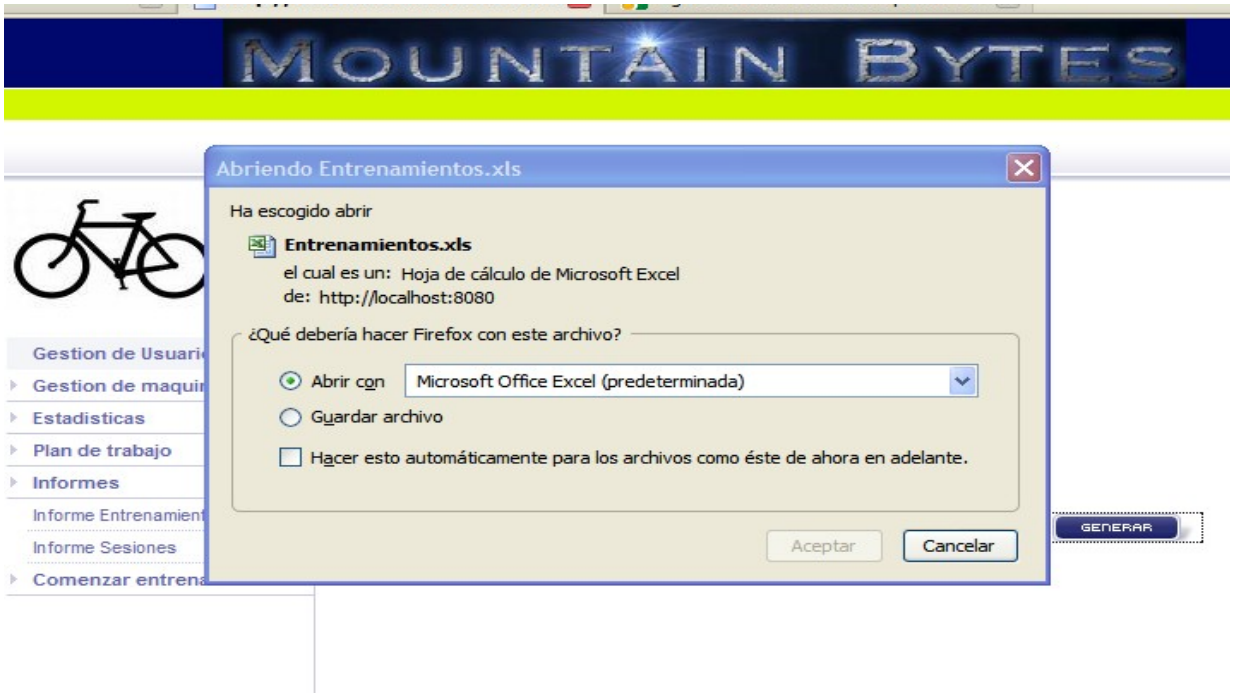


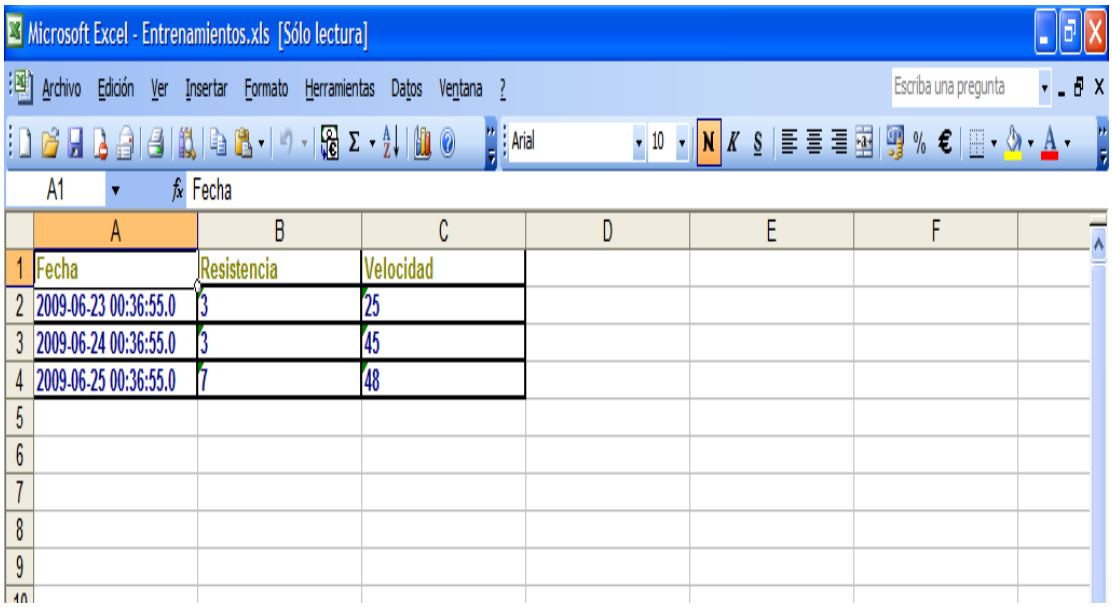
El siguiente módulo es el correspondiente a informes. Este módulo permite obtener y exportar informes de sesiones y entrenamientos en formato Excel (XLS). El funcionamiento de este módulo es muy sencillo, solamente hay que seleccionar si deseamos un informe de entrenamientos o sesiones, y seleccionar el intervalo de tiempo que queremos que abarque el informe. Podemos o bien introducir la fecha a mano en el campo correspondiente, o bien pulsar en el icono que hay en el lateral y que permite seleccionar el intervalo de fechas desde un sencillo calendario. Deberemos también elegir qué parámetros queremos que aparezcan en el informe: velocidad, resistencia o ambos. Cuando se pulse en “generar” el

informe deseado se creará automáticamente, y el navegador iniciará su descarga a nuestro ordenador.

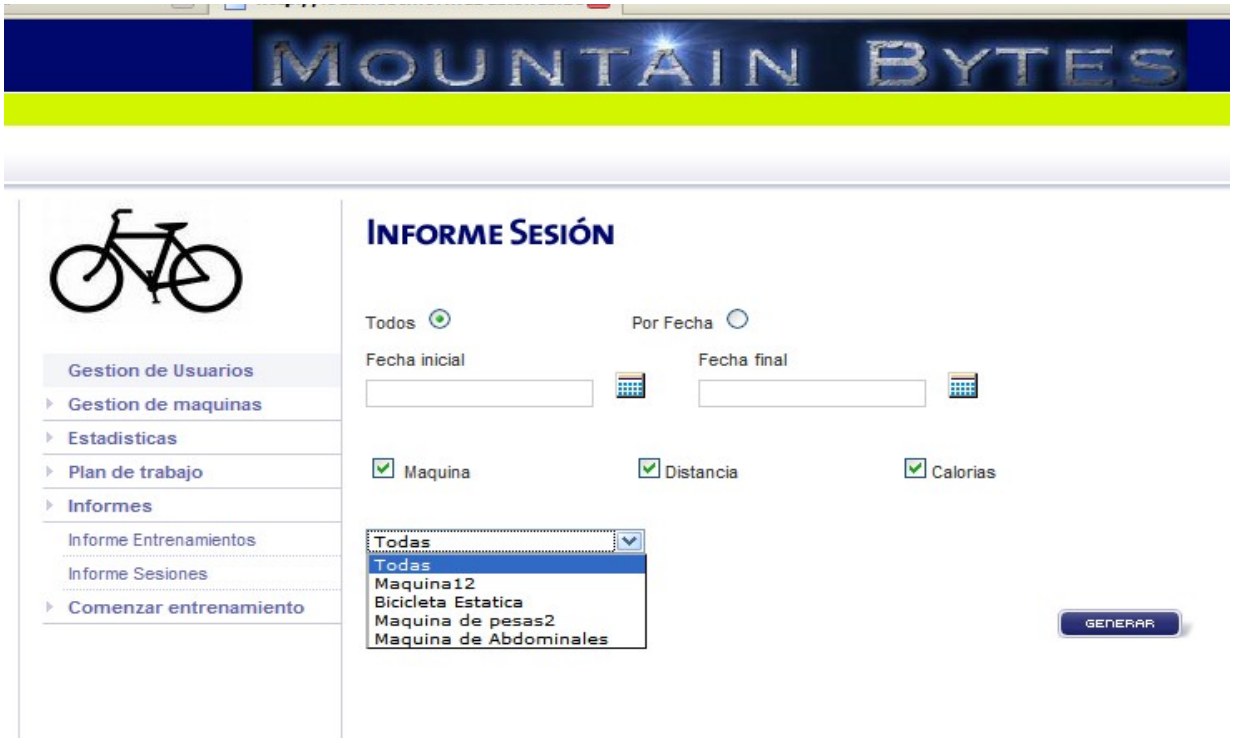
A través de esta funcionalidad es usuario podrá tener informes del trabajo que ha realizado durante el periodo deseado. Puede ver todo lo que ha hecho o reducir a un rango de fechas seleccionado. Si pulsamos en los calendarios aparece un calendario para seleccionar las fechas, tras pinchar en la fecha deseada el calendario desaparece automáticamente y la fecha seleccionada se sitúa en el campo fecha.

Pulsando en “generar” obtendremos:

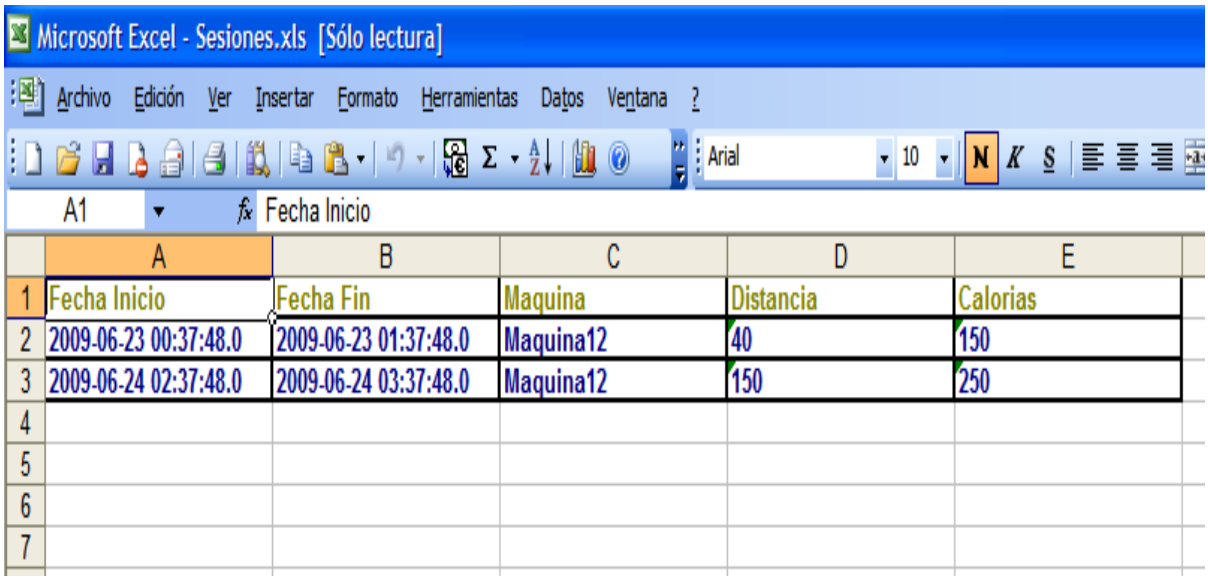




El informe de sesiones es muy similar; a través de esta funcionalidad el usuario puede ver los informes relativos a sus sesiones. Especificando no sólo si desea un rango de fechas como antes, sino también si quiere ver sus sesiones referidas a todas las máquinas o sólo a una específicamente.

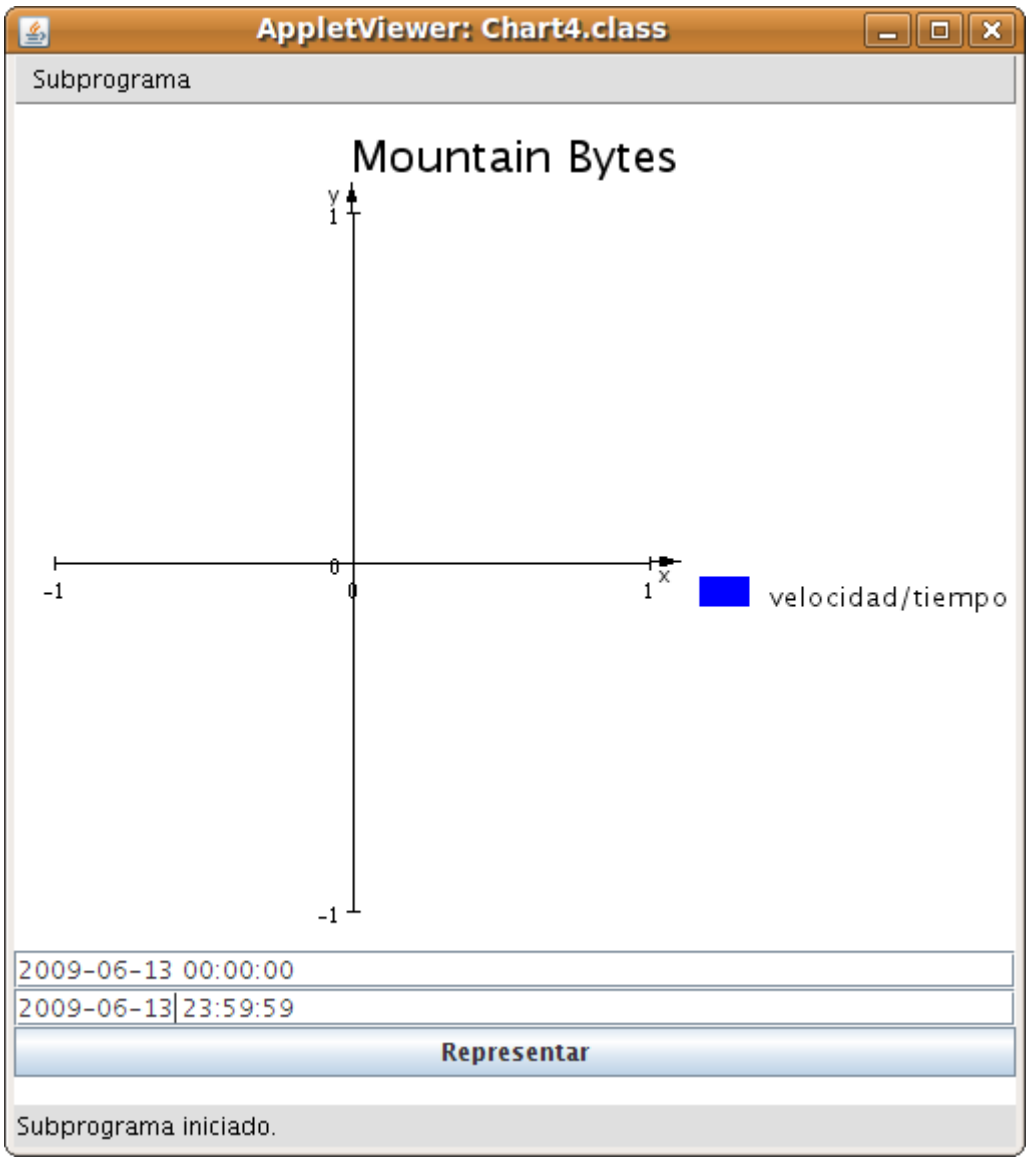


Como resultado aparece la misma ventana de antes para seleccionar con que programa deseamos abrir la información y tras abrirlo nos encontraremos con algo parecido a esto.

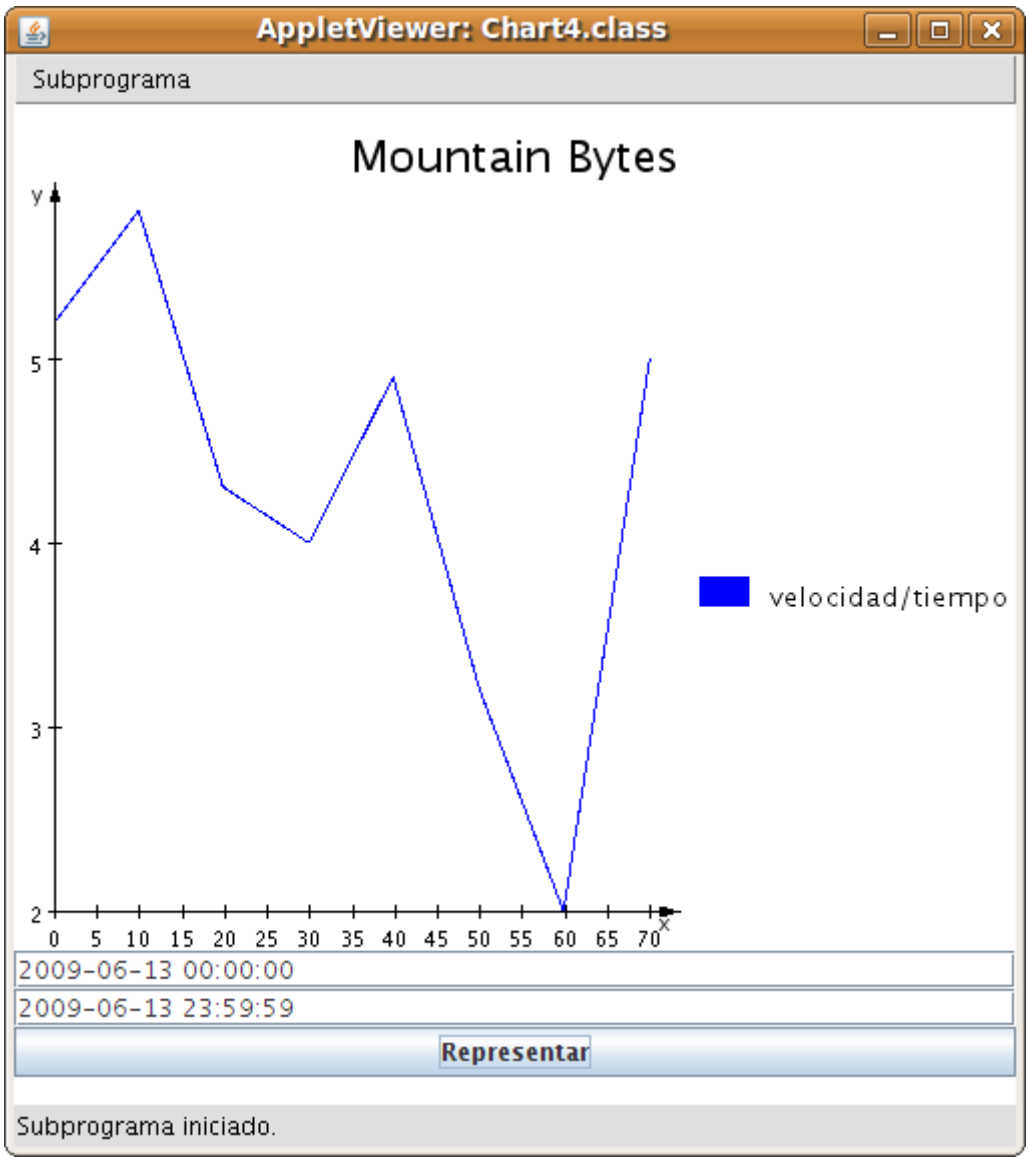


	A	B	C	D	E
1	Fecha Inicio	Fecha Fin	Maquina	Distancia	Calorias
2	2009-06-23 00:37:48.0	2009-06-23 01:37:48.0	Maquina12	40	150
3	2009-06-24 02:37:48.0	2009-06-24 03:37:48.0	Maquina12	150	250
4					
5					
6					
7					

En cuanto a la parte de estadísticas implementada a través de un applet Java, después de su inicialización encontraremos que la interfaz nos permite indicar el rango de fechas que queremos ver:

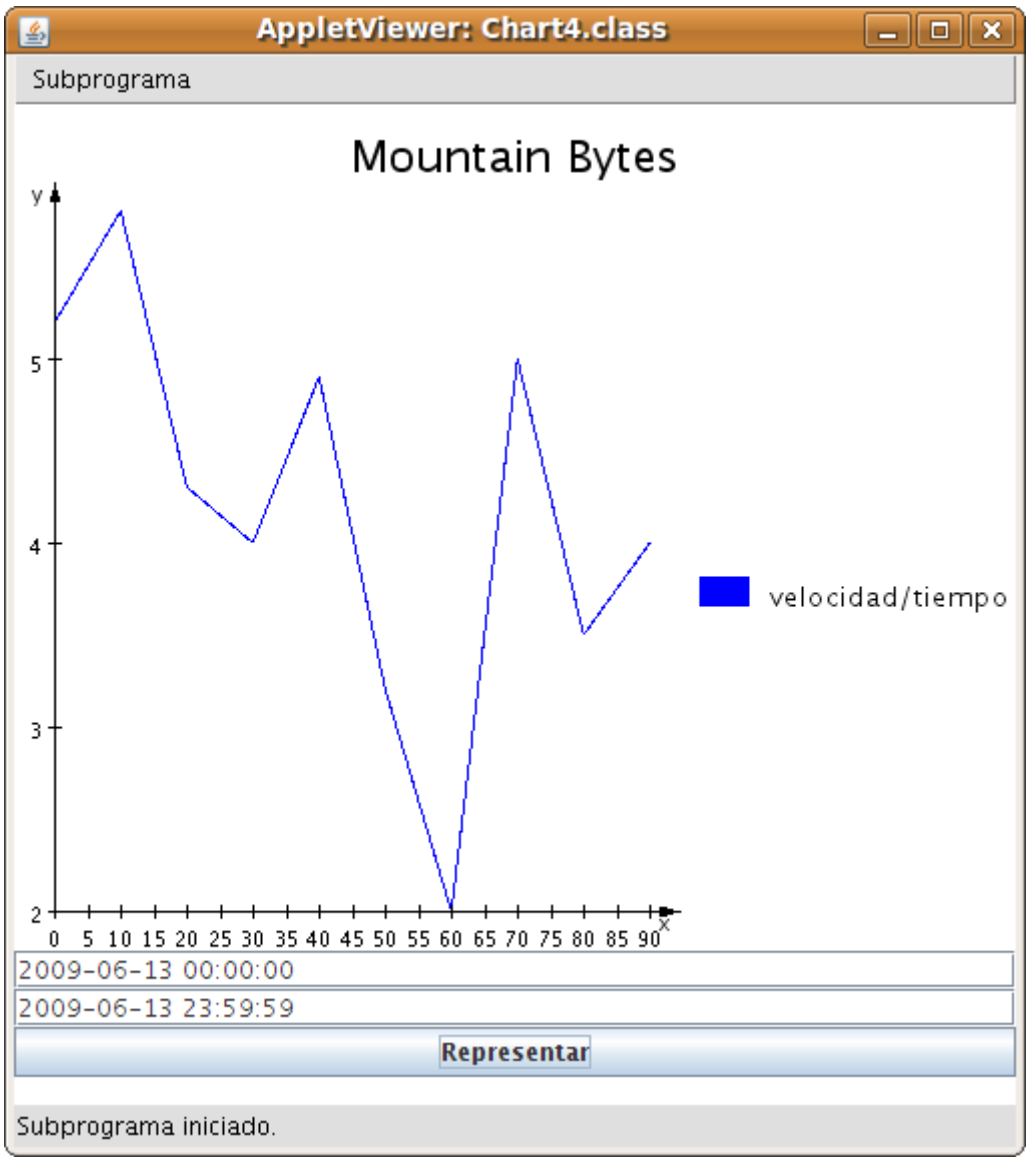


Una vez definido el intervalo, pulsaremos en “representar” para que se dibuje el gráfico. En este instante los datos de actividad relativos al periodo indicado se recogerán de la base de datos y se mostrarán en el gráfico que aparecía vacío en la primera imagen:



En este ejemplo había ocho valores en la base de datos. Como podemos ver, el eje de la velocidad comienza en el menor valor existente y no en cero. El eje de tiempo representa fechas concretas, pero a efectos de representación han sido convertidas a segundos desde la primera entrada encontrada en la base de datos, que corresponderá al valor cero.

Dado que el applet se refresca automáticamente en un corto periodo de tiempo, después de esperar 20 segundos podemos observar que hay dos entradas más, correspondientes a datos que acaban de ser registrados. El fin de este refresco automático es poder representar lo más fielmente posible la actividad real en una máquina, y poder disponer de datos casi en tiempo real si estamos consultando esta gráfica a la vez que utilizamos una máquina.

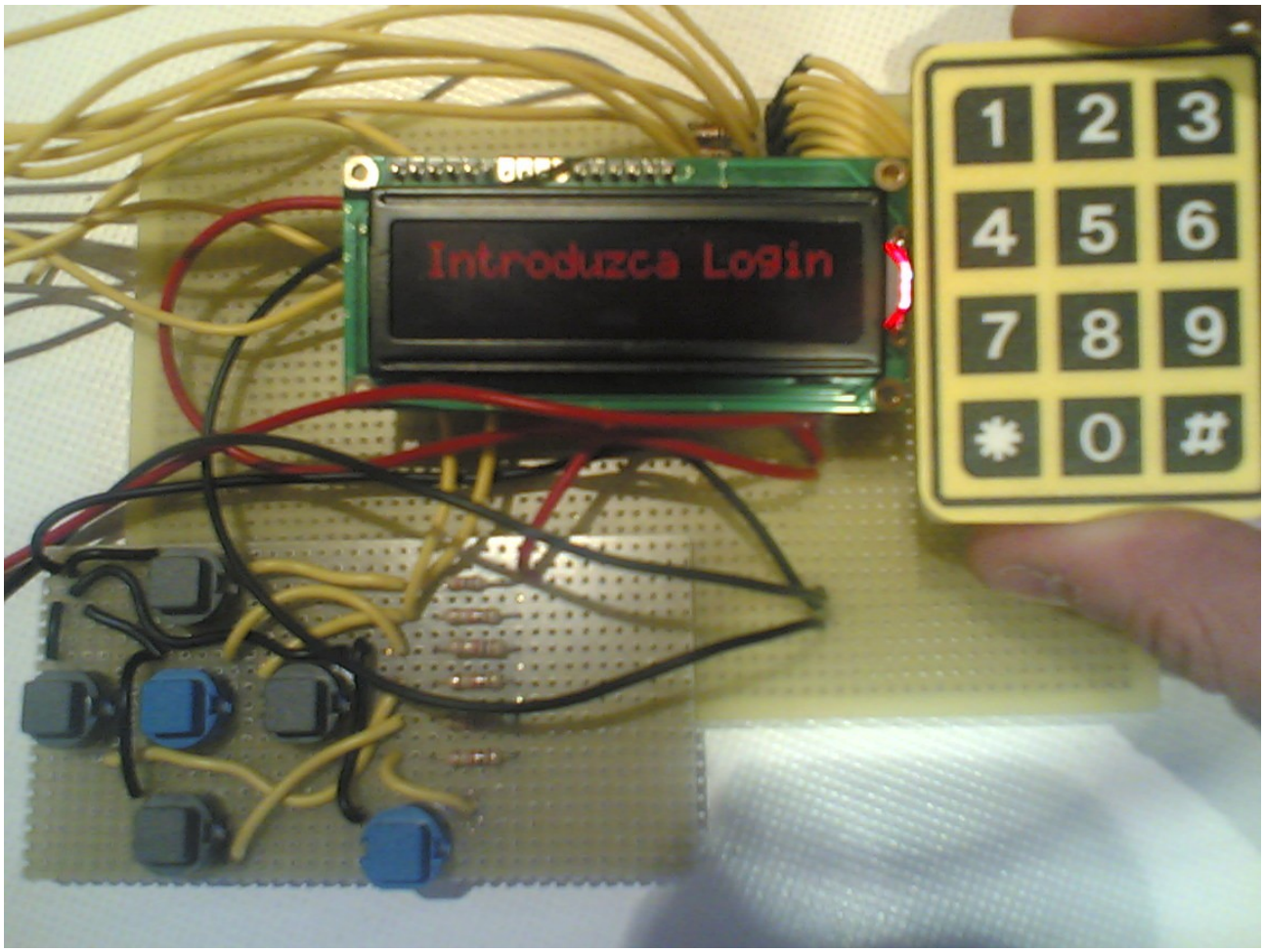


Interfaz de Arduino

La interfaz del módulo hardware es mucho más sencilla que la de la aplicación que se ejecuta en el servidor. La placa Arduino consta de cuatro botones de dirección para ajustar el esfuerzo cuando el ejercicio ha comenzado, un botón para cancelar la operación en curso en cualquier momento, y un botón de aceptar. Consta así mismo de un teclado numérico para introducir la información de login y otros datos que pueda pedir la aplicación, como por ejemplo el peso del usuario, que luego se enviarán al servidor, se almacenarán y podrán utilizarse para el cálculo de estadísticas o como registro del progreso del usuario.

La forma de mostrar datos al usuario es mediante un display que imprime información o bien preguntas dirigidas al usuario.

La primera pantalla que muestra el módulo hardware es la de login; el usuario debe introducir su identificador y éste será validado en el servidor. Si no es correcto el display mostrará un mensaje de error y no permitirá el acceso; en caso contrario, el usuario pasará a la fase de selección de tipo de ejercicio.



En la pantalla de selección del ejercicio, el usuario puede indicar si desea realizar un plan de entrenamiento determinado con anterioridad en la aplicación web por él o por un administrador; o bien si desea realizar un entrenamiento libre, que también será monitorizado.

Si el usuario eligió un entrenamiento predefinido, el display indicará al usuario qué niveles de velocidad y esfuerzo debe utilizar en dicho entrenamiento. Si por el contrario el usuario eligió un entrenamiento libre, podrá indicar a la máquina cuál es el nivel de esfuerzo que está utilizando en cada momento del ejercicio mediante los pulsadores de dirección.

En cualquier momento, el usuario puede abandonar el ejercicio pulsando el botón de cancelación, que finaliza inmediatamente el entrenamiento y retorna la máquina al estado de login.

Apéndice I: terminología

A continuación hacemos una reseña sobre términos frecuentes en esta documentación, separando los que pertenecen a la parte hardware de los que pertenecen al software.

Hardware

Bluetooth

Protocolo inalámbrico para la transmisión de información mediante radiofrecuencia. Permite la creación de redes de dispositivos denominadas "personal area networks" y permite la conexión con varios dispositivos simultáneamente.

Capa física

Es la primera capa en el modelo OSI de redes. Especifica la interconexión a nivel físico de una red. En esta capa se define la forma de transmitir bits; entre otras cosas, modulación, frecuencias, cableado...

Capa MAC (media access control)

Está integrada en la capa de enlace de datos del modelo OSI. Especifica cómo se controla el acceso a una red.

Wi-Fi

Protocolo de conexión para redes inalámbricas de radiofrecuencia basado en el estándar IEEE 802.11. Es el estándar "de facto" para redes inalámbricas de computadores personales

y otros productos de electrónica de consumo tales como videoconsolas, impresoras inalámbricas o teléfonos móviles de gama alta. Existe un organismo, la "WiFi Alliance", que certifica los diversos dispositivos Wi-Fi para confirmar que efectivamente cumplen con los estándares requeridos.

Software

Applet

Un applet Java es código escrito en el lenguaje de programación Java y compilado en bytecode (ver más abajo) que se descarga en el navegador web de un cliente y se ejecuta en su computador, en lugar de ejecutarse en el servidor. Los applet permiten típicamente la representación gráfica de información, la consulta a bases de datos, y casi cualquier otra acción que pueda implementarse en una aplicación Java estándar, sien do la diferencia principal que el applet se ejecuta en el navegador del cliente.

Bytecode

Dado que uno de los objetivos del lenguaje Java es la portabilidad, la solución de compromiso que se eligió entre diseñar un nuevo lenguaje compilado o interpretado fue la de crear un formato intermedio, independiente de la arquitectura en la que se ejecuta, que no obstante tiene muchas similitudes con los lenguajes de tipo ensamblador y que por ello es más sencillo de interpretar que un lenguaje interpretado clásico. Este formato intermedio se denomina bytecode, y para su ejecución es necesario tener instalada la máquina virtual Java, que es el programa que traslada este lenguaje intermedio a instrucciones reales de la máquina en la que está instalado.

JavaScript

JavaScript es un lenguaje interpretado usado fundamentalmente para la generación de contenido dinámico en la web, ya que está soportado por la mayoría de navegadores importantes. A pesar de su nombre deriva del estándar ECMAScript, no del lenguaje Java, si bien ambos comparten una sintaxis esencialmente inspirada en C.

JSP

JSP o JavaServer Pages es una tecnología que permite insertar código escrito en el lenguaje de programación Java dentro de páginas web diseñadas en HTML o XHTML. Este código debe ser escrito dentro de etiquetas especiales que lo delimitan y que indican al servidor web qué partes debe ejecutar como código Java. Posteriormente este código se compila y se convierte en un objeto ejecutable llamado servlet. No todos los servidores web permiten el uso de JSPs.

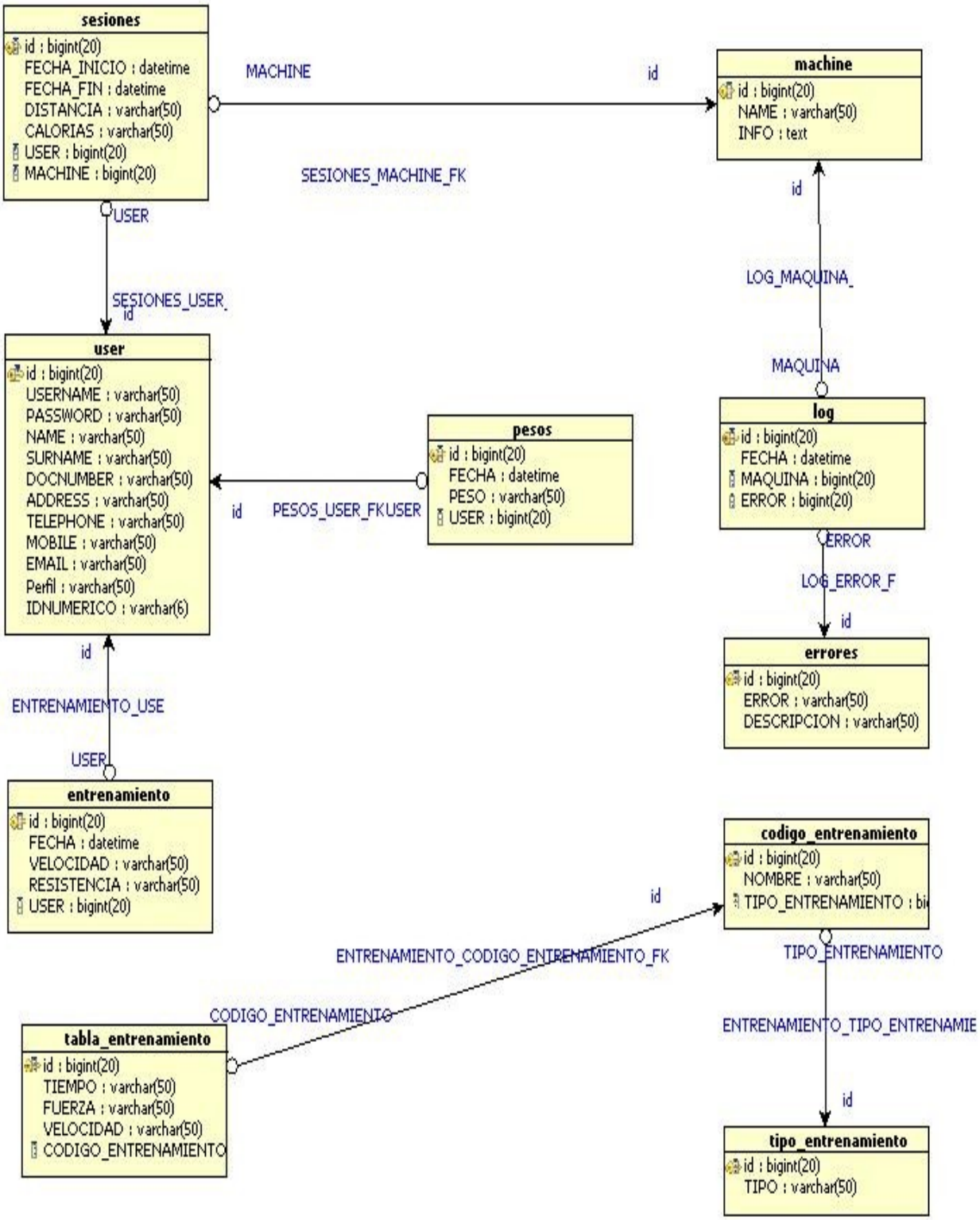
Servidor web

Programa que se ejecuta en un computador y que sirve contenido, usualmente en formato HTML o XHTML, a múltiples clientes, que se conectan a este servidor a través de otro programa llamado navegador.

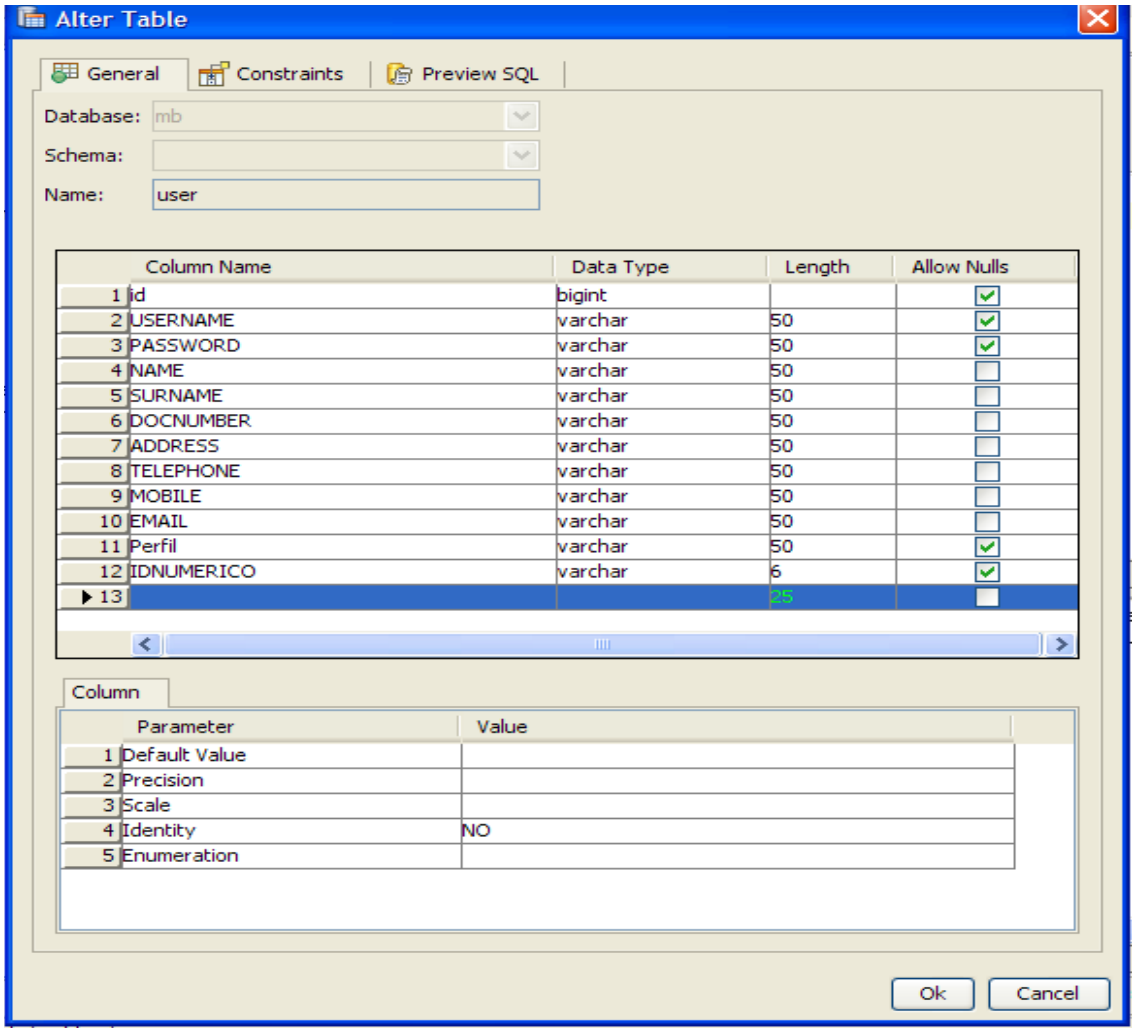
Servlet

El código escrito en Java/JSP se compila y se convierte en un servlet. Ver “JSP” más arriba.

Apéndice II: modelo de la base de datos



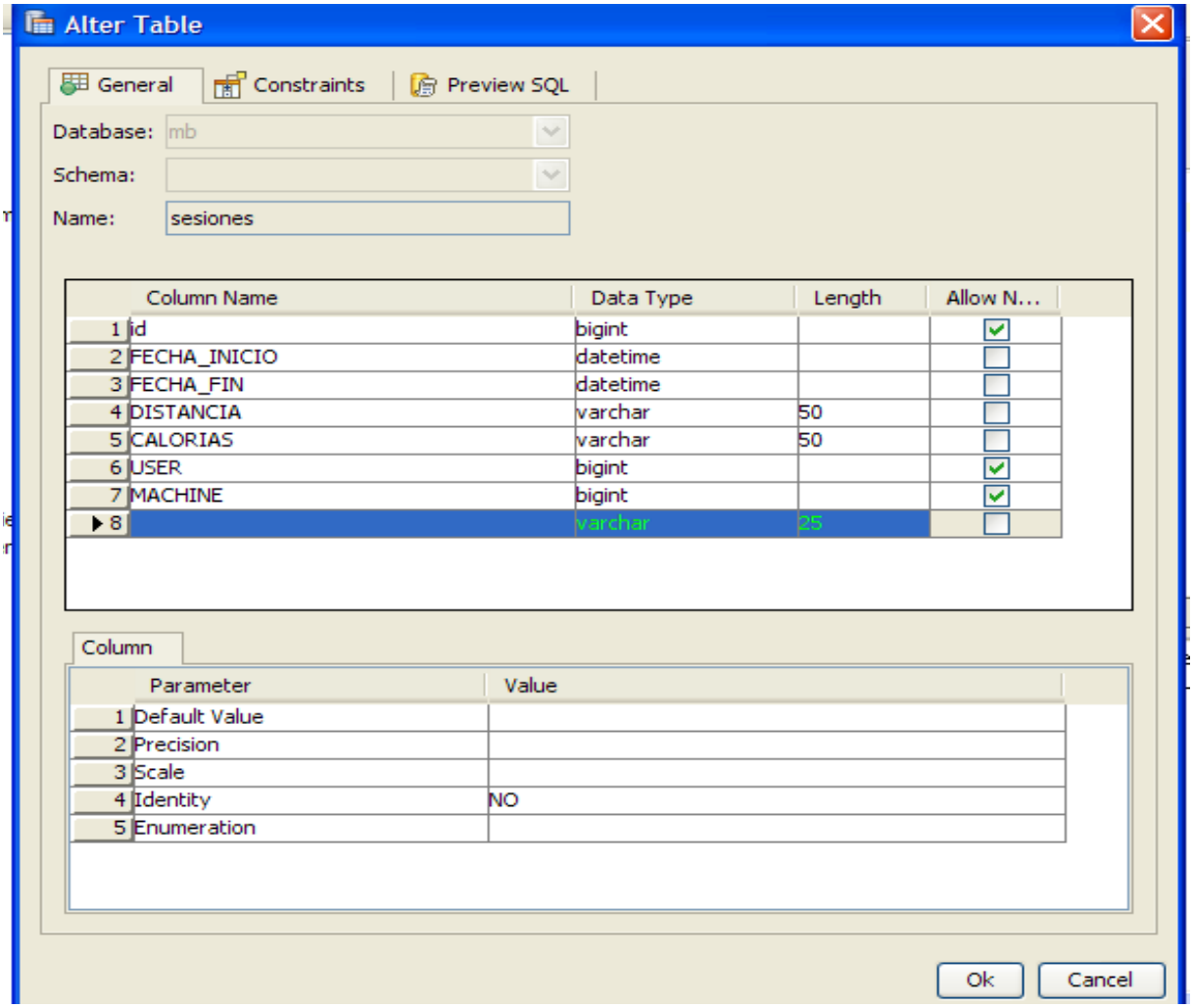
User



- **Id**: representa el autonumérico de la table, es su clave primaria
- **Username**: representa el identificador del usuario necesario para la entrada a la aplicación.
- **Password**: es la clave de entrada al sistema. Inicialmente se inicializa con el DNI del usuario.
- **Name**: nombre del usuario.
- **Surname**: representa el apellido del usuario.
- **DocNumber**: es el DNI del usuario
- **Address**: el la dirección del usuario.
- **Telephone**: es el número de teléfono del usuario.
- **Mobile**: representa el móvil del usuario.

- **Email**: es la dirección de correo electrónico del usuario.
- **Perfil**: describe el perfil del usuario. Si su valor el 0 implica que el usuario es un administrador del sistema por lo que tiene habilitadas más funciones en la aplicación, si por el contrario este campo toma el valor 1 implica que es un usuario normal por lo que hay funcionalidades que no le aparecerán en la aplicación.
- **IdNumerico**: es un identificador asignado al usuario, es la clave de acceso a la máquina de entrenamiento. Este identificador no puede estar repetido, de ello de encarga la aplicación a la hora de su creación.

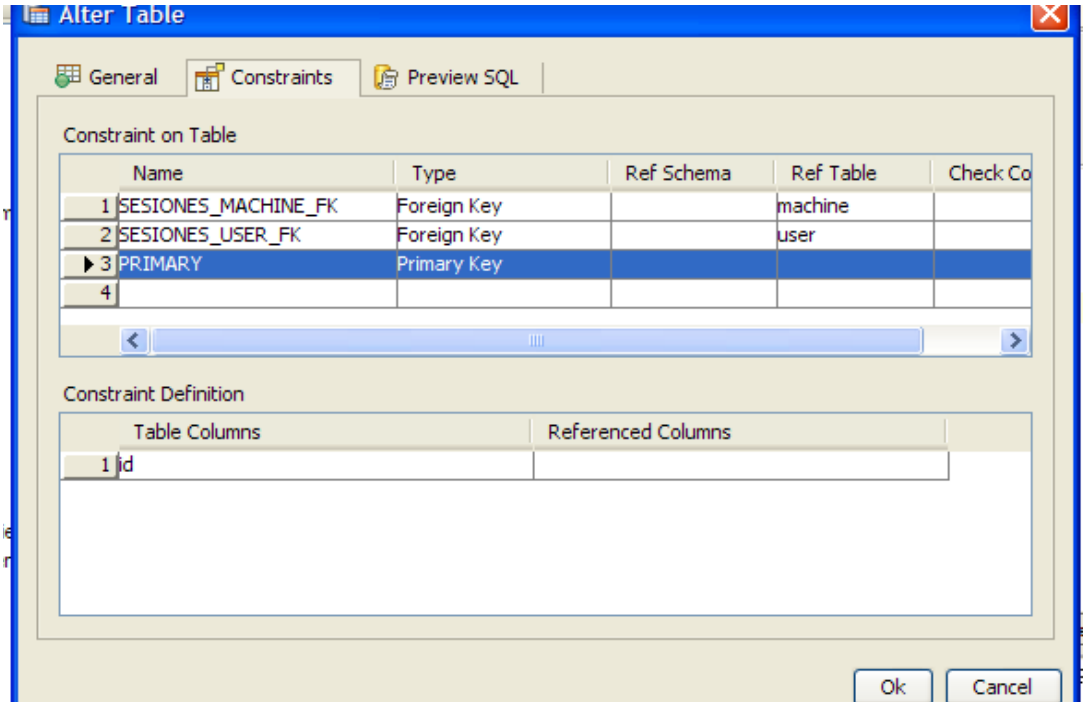
Sesiones



- **Id**: representa el autonumérico de la table, su clave primaria
- **Fecha_Inicio**: representa la fecha, hora, minutos y segundos en el que se empezó la

sesión.

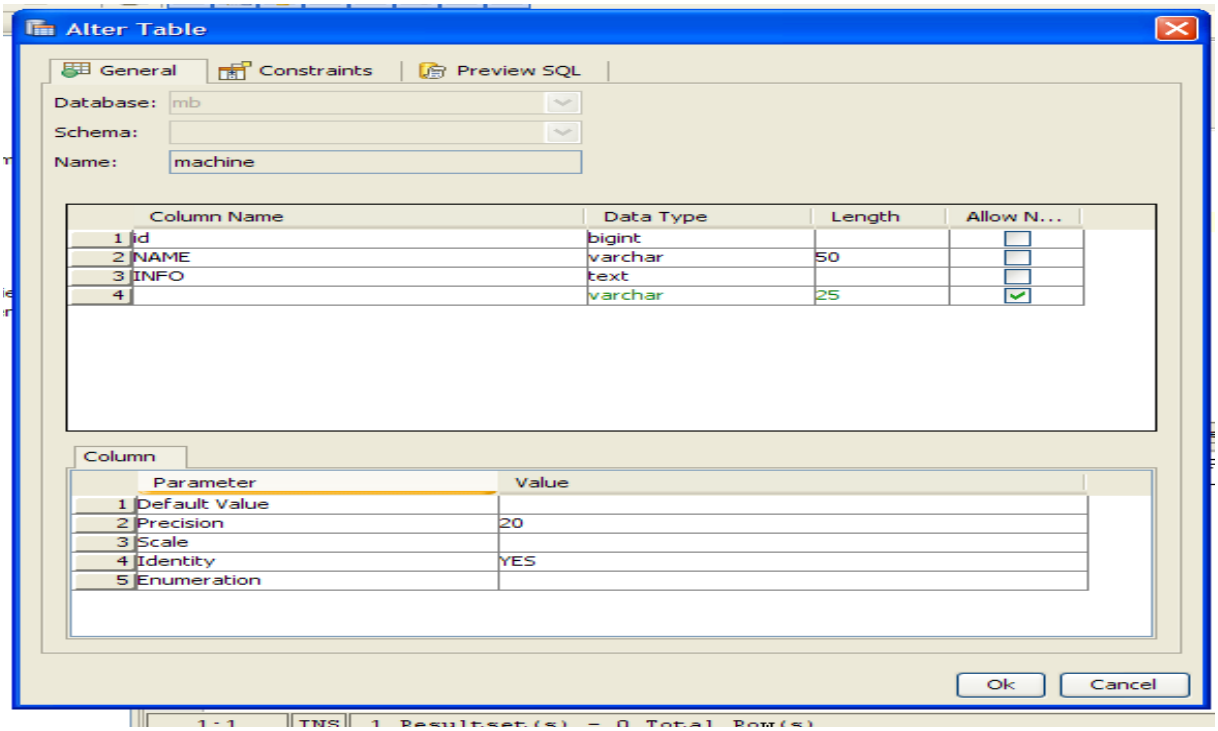
- **Fecha_Fin**: representa la fecha, hora, minutos y segundos en el que se finalizó la sesión.
- **Distancia**: es el cálculo de la distancia recorrida durante la sesión.
- **Calorías**: representa las calorías consumidas durante el entrenamiento.



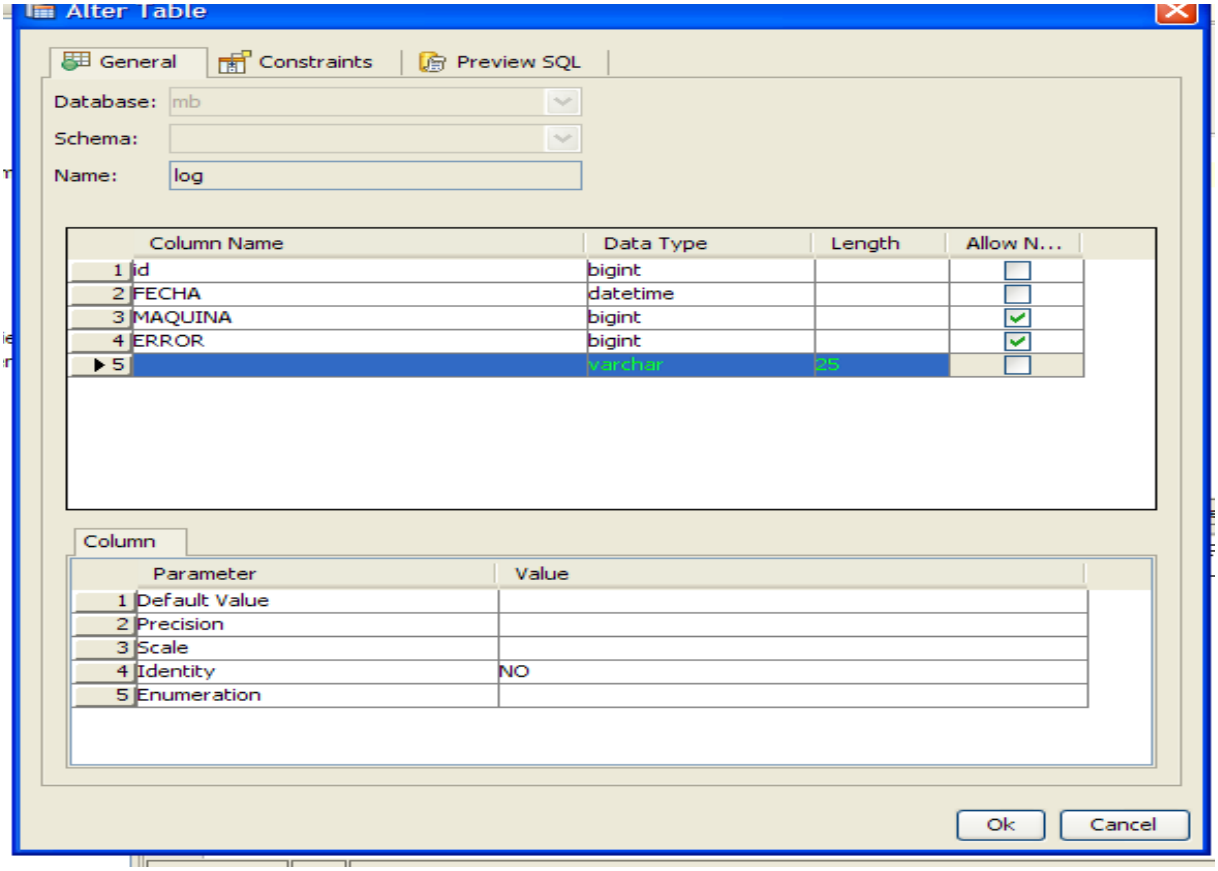
- **User**: es una foreign key, hace referencia a la tabla de usuarios de esta manera cada sesión queda relacionada con el usuario que la realizó.
- **Machine**: es una foreign key hace referencia a la tabla de máquinas, de esta forma en cada sesión podemos saber en qué máquina se realizó.

Machines

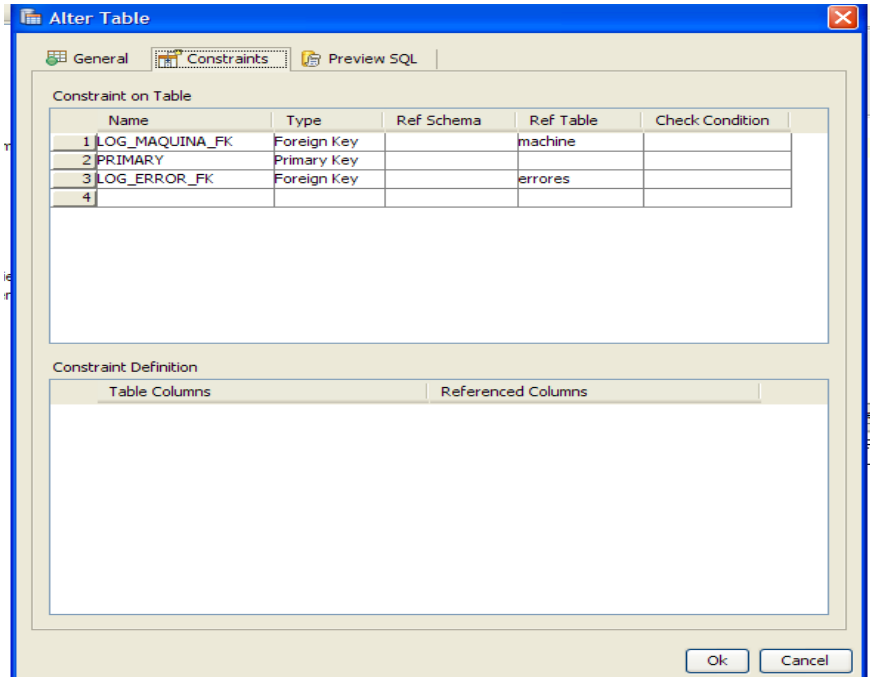
- **Id**: representa el autonumérico de la tabla, es su clave primaria.
- **Name**: es el nombre de la máquina.
- **Info**: contiene información adicional de la máquina.


Alter Table
General
Constraints
Preview SQL
Database: mb
Schema:
Name: machine
Column Name
Data Type
Length
Allow N...
1 id bigint
2 NAME varchar 50
3 INFO text
4 varchar 25
Column
Parameter
Value
1 Default Value
2 Precision 20
3 Scale
4 Identity YES
5 Enumeration
Ok
Cancel

Log

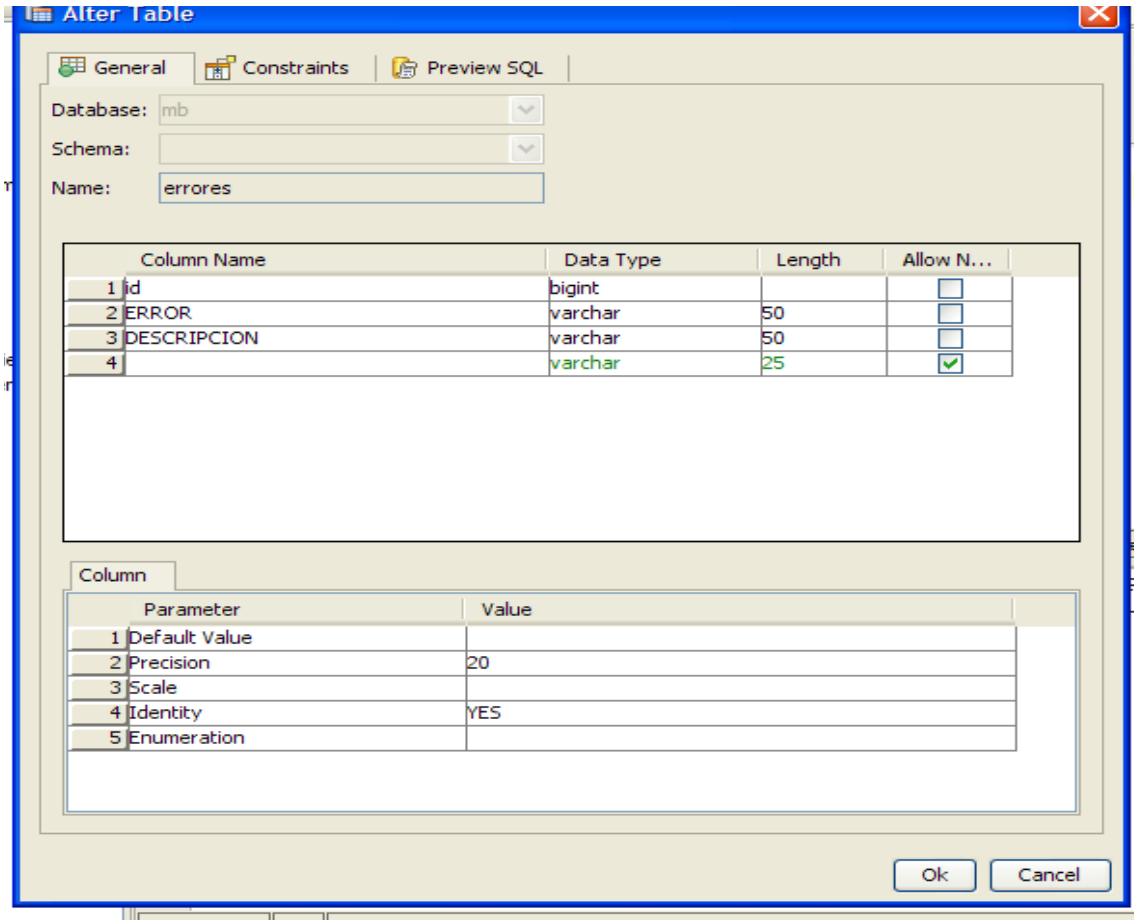

Alter Table
General
Constraints
Preview SQL
Database: mb
Schema:
Name: log
Column Name
Data Type
Length
Allow N...
1 id bigint
2 FECHA datetime
3 MAQUINA bigint
4 ERROR bigint
5 varchar 25
Column
Parameter
Value
1 Default Value
2 Precision
3 Scale
4 Identity NO
5 Enumeration
Ok
Cancel

- **Id**: representa el autonumérico de la tabla, es su clave primaria.
- **Fecha**: es la fecha en la que se realizó el error.
- **Maquina**: es una foreign key hace referencia a la tabla de máquinas para saber en que máquina se presentó el error.
- **Error**: es una foreign key hace referencia a la tabla de errores para saber qué error fue el que se dio.



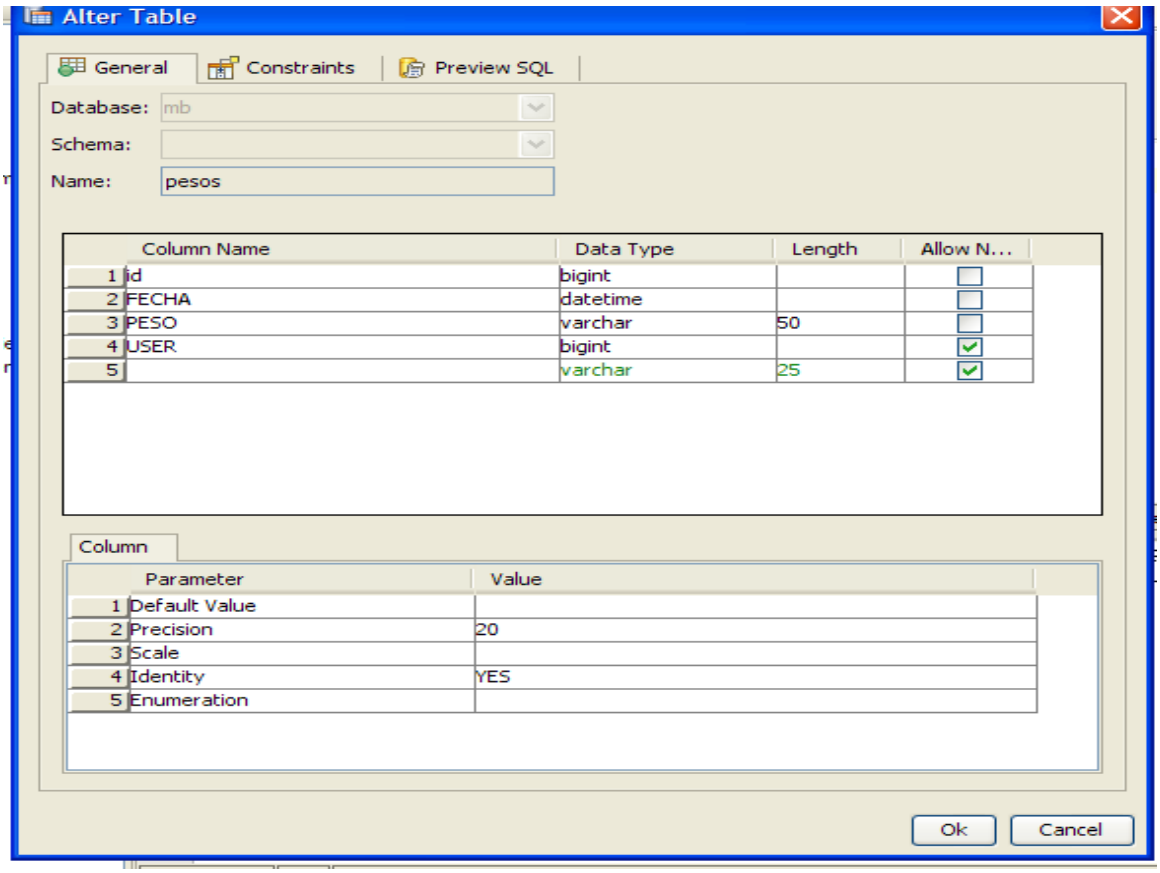
Errores

- **Id**: representa el autonumérico de la tabla, es su clave primaria.
- **Error**: nombre del error.
- **Descripción**: descripción del error.



Pesos

- **Id**: representa el autonumérico de la tabla, es su clave primaria.
- **Fecha**: representa la fecha en la que el peso del usuario era el que se refleja en el campo peso.
- **Peso**: refleja el peso del usuario en la fecha indicada.
- **User**: es una foreign key hace referencia a la tabla de usuarios para saber que usuario es el que tiene dicho peso.


Alter Table
General
Constraints
Preview SQL
Database: mb
Schema:
Name: pesos
Column Name
Data Type
Length
Allow N...
1 id bigint
2 FECHA datetime
3 PESO varchar 50
4 USER bigint
5 varchar 25
Column
Parameter
Value
1 Default Value
2 Precision 20
3 Scale
4 Identity YES
5 Enumeration
Ok
Cancel

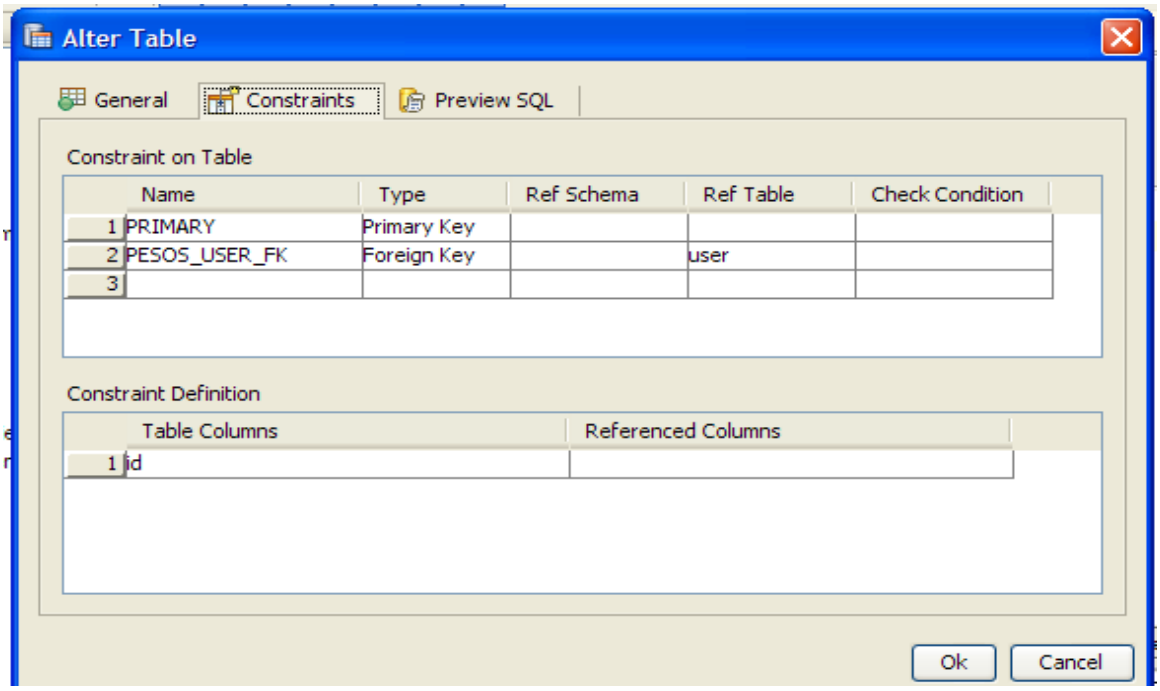
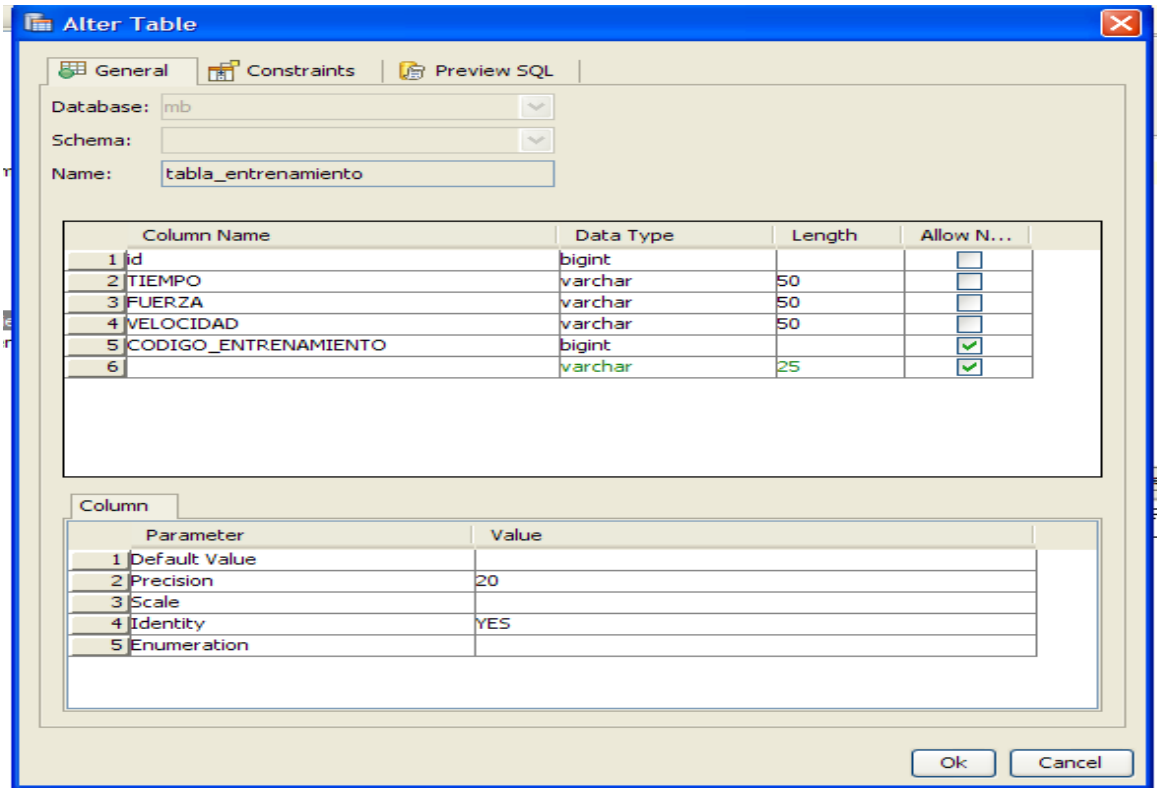
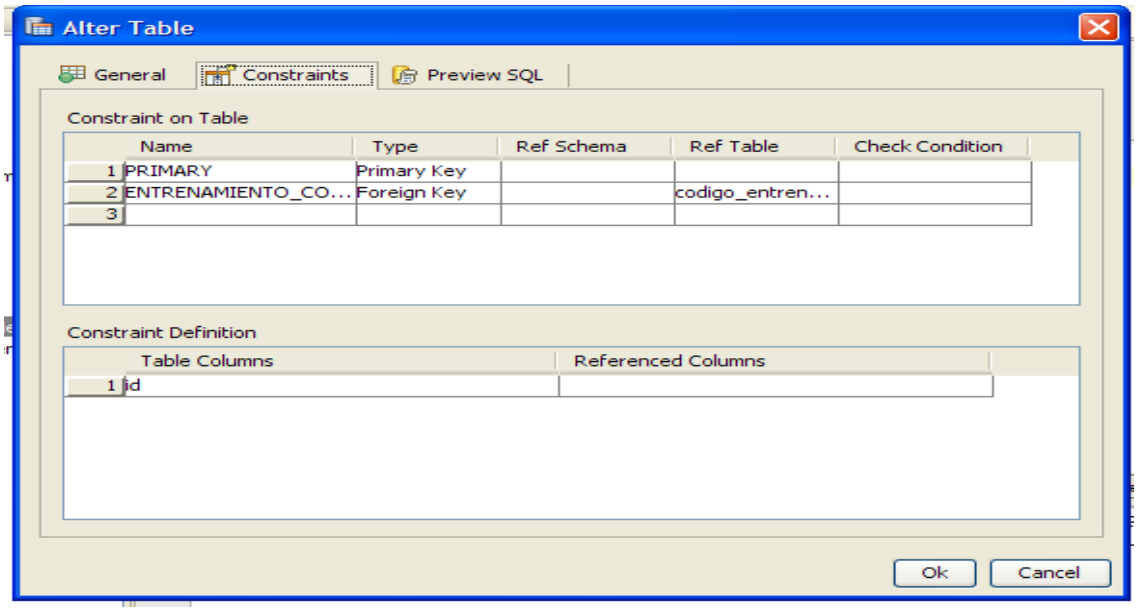

Alter Table
General
Constraints
Preview SQL
Constraint on Table
Name
Type
Ref Schema
Ref Table
Check Condition
1 PRIMARY Primary Key
2 PESOS_USER_FK Foreign Key user
3
Constraint Definition
Table Columns
Referenced Columns
1 id
Ok
Cancel

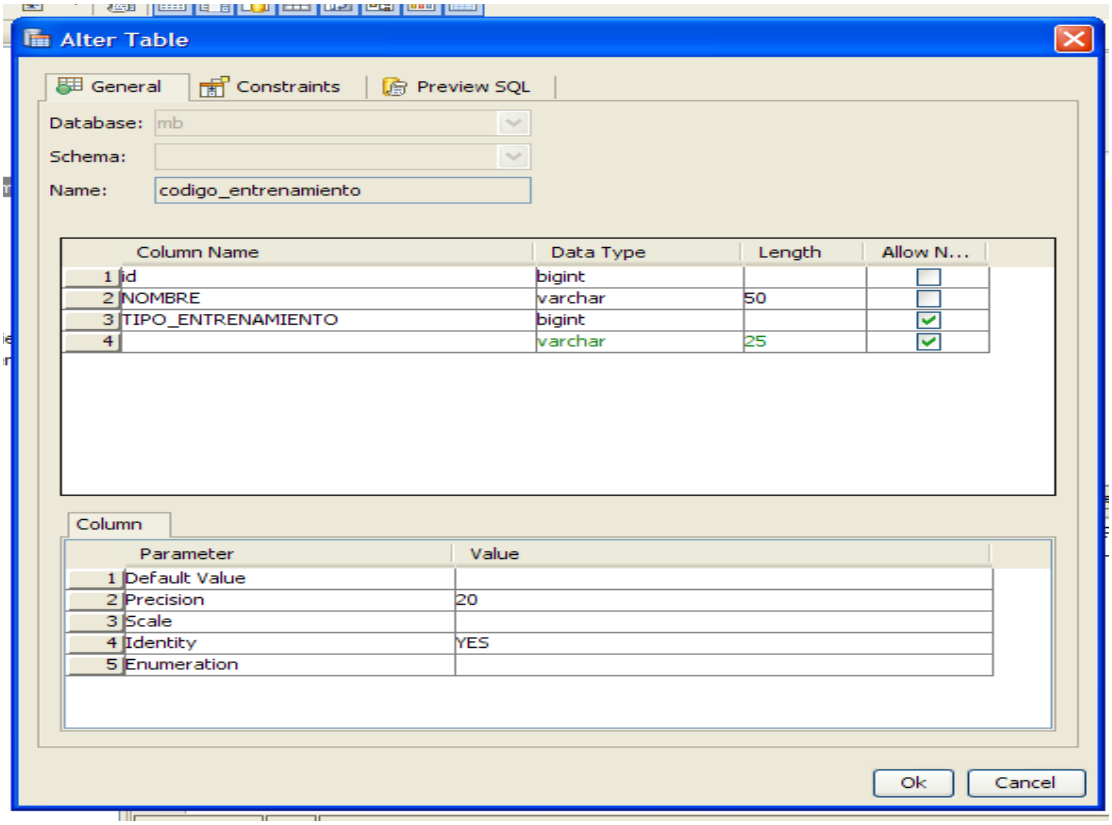
Tabla Entrenamientos



- **Id**: representa el autonumérico de la tabla, es su clave primaria.
- **Tiempo**: tiempo en minutos que dura el entrenamiento con estas características de fuerza y velocidad.
- **Fuerza**: fuerza del entrenamiento.
- **Velocidad**: representa la velocidad en Km/h.
- **Codigo_Entrenamiento**: es una foreign key hace referencia a la tabla código del entrenamiento para saber cuál es el entrenamiento que se está realizando.

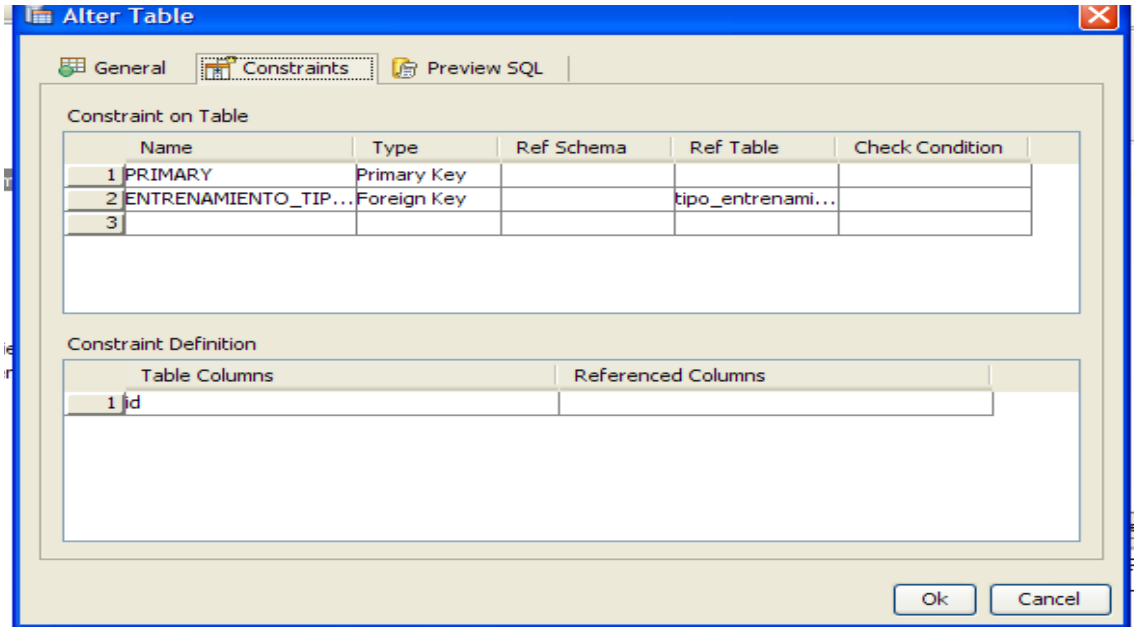


Código Entrenamiento

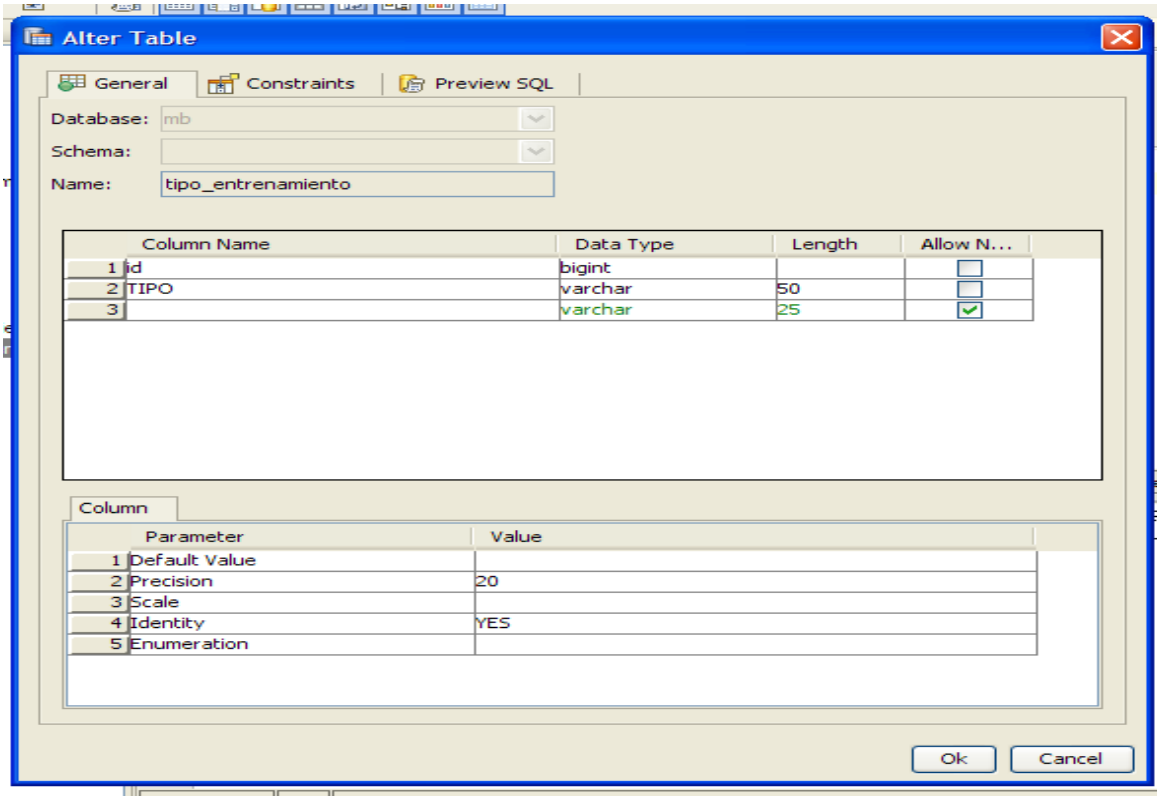


- **Id**: representa el autonumérico de la tabla, es su clave primaria.
- **Nombre**: es el nombre del entrenamiento.

- **Tipo_entrenamiento**: es una foreign key hace referencia a la tabla tipo de entrenamiento para saber de que tipo de entrenamiento de trata.

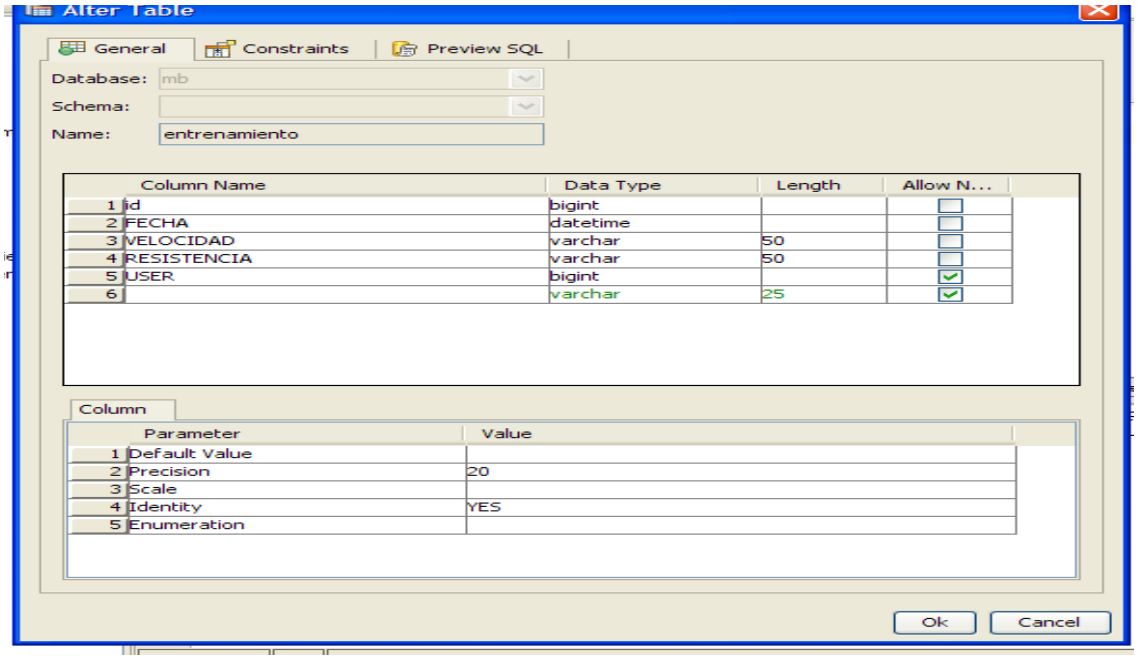


Tipo Entrenamiento

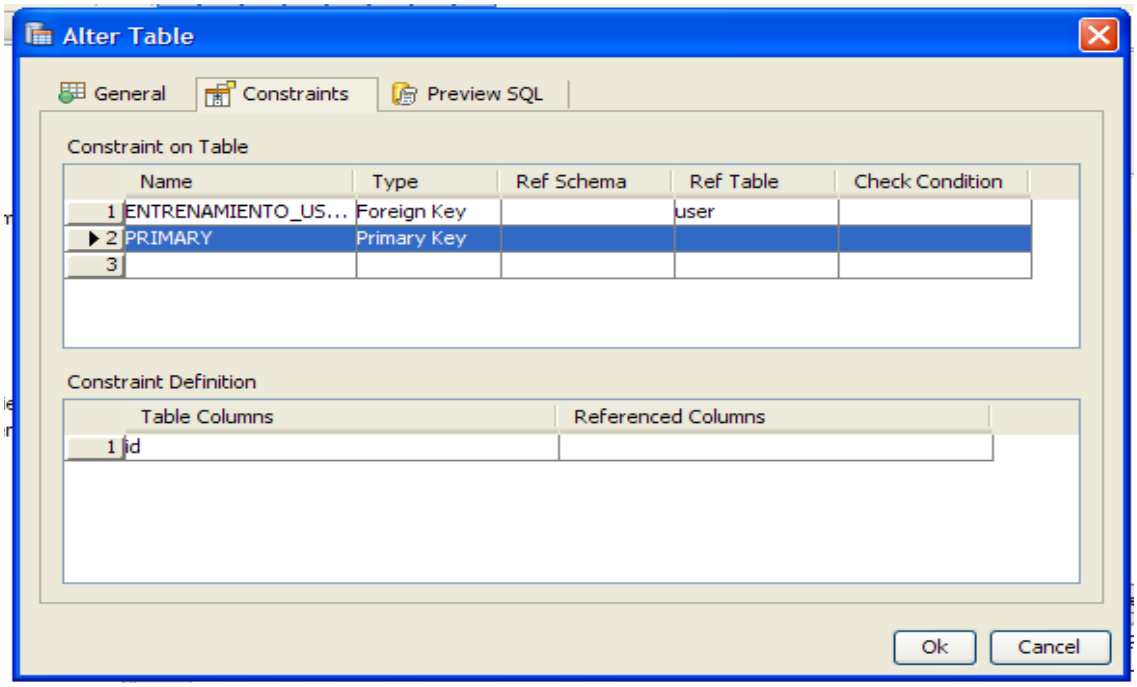


- **Id**: representa el autonumérico de la tabla, es su clave primaria.
- **Tipo**: es el tipo del entrenamiento.

Entrenamiento



- **Id**: representa el autonumérico de la tabla, es su clave primaria.
- **Fecha**: representa la fecha en la que el usuario realizó el entrenamiento.
- **Velocidad**: es la velocidad a la que va el usuario.
- **Resistencia**: representa la resistencia que presenta el usuario.
- **User**: es una foreign key hace referencia a la tabla de user para saber a qué usuario pertenece el entrenamiento.


Alter Table
General
Constraints
Preview SQL
Constraint on Table
Name
Type
Ref Schema
Ref Table
Check Condition
1 ENTRENAMIENTO_US...
Foreign Key
user
2 PRIMARY
Primary Key
3
Constraint Definition
Table Columns
Referenced Columns
1 id
Ok
Cancel

Apéndice III: estructura de la aplicación

src/main/java

- **es.mountainBytes.beans:** paquete que contiene los beans, en nuestro caso los beans de negocio y de la vista son los mismos, por lo que en vez de duplicar código usamos el mismo bean indistintamente para las vistas que para los servicios y daos.
- **es.mountainBytes.comunicacion:** en este paquete está implementado todo lo relacionado con la comunicación con Arduino.
- **es.mountainBytes.dao:** paquete que contiene las interfaces de la parte referente a la base de datos.
- **es.mountainBytes.daoImpl:** paquete que hace referencia a la parte de persistencia, mediante las clases que contiene sacamos datos de la base de datos o los introducimos.
- **es.mountainBytes.service:** paquete que contiene las interfaces referentes a los servicios.
- **es.mountainBytes.serviceImpl:** paquete que contiene las clases que implementas los diferentes servicios, tratan los datos recibidos por el controlador haciendo las operaciones necesarias y de necesitar parte de persistencia hacen las llamadas a los distintos dao.
- **es.mountainBytes.web:** las clases que contienen se encargan tanto de la gestión de las distintas conexiones como de las transacciones de la base de datos.
- **es.mountainBytes.web.controller:** contiene los distintos controladores encargados de las llamadas a las vistas y de la recepción de datos por parte de las mismas. Almacenan los datos en las estructuras de datos correspondientes y en caso de necesitar tratarlos llaman a los servicios implicados.
- **es.mountainBytes.web.reports:** contiene las clases encargadas de realizar los archivos de informes en formato Excel.

- **es.mountainBytes.web.validation:** contienen las clases que validan los datos en le jsp, comprueban que los datos recibidos por el controlador tienen las propiedades esperadas.

src/main/resources

- **hibernate.cfg.xml:** archivo de configuración de Hibernate.
- **hibernate.properties:** archivo en formato de texto plano que contiene las variables necesarias para la configuración de la conexión a la base de datos a través de Hibernate.
- **X.hbm.xml:** los distintos archivos de configuración para el mapeo de los beans con las distintas tablas de la base de datos. Necesarios por hibernate para la realización de las queries.
- **messages.properties:** archivo que contiene los distintos mensajes informativos a mostrar por los jsp.
- **configurer.properties:** archivo de configuración de las distintas variables utilizadas en el jsp.
- **views.properties:** archivo de configuración de las distintas vistas, en este archivo se realiza el mapeo del nombre de la vista con la dirección del jsp que resuelve dicha vista.

src/main/resources/webapp

- **css:** contiene los distintos css, hojas de estilo para los jsp.
- **images:** contiene las distintas imágenes utilizadas por los jsp.
- **js:** contiene los distintos archivos .js donde están implementadas las funcionalidades de java-Scripts.

src/main/resources/web-inf

- **include:** contiene ficheros .jsp que se usan muy a menudo.
- **jsp:** carpeta que contiene los distintos jsp usados por la aplicación.
- **tld:** contiene los distintos tld que ofrece Spring.
- **applicationContext.xml:** fichero de configuración del contexto de Spring.
- **mountainBytes-servlet.xml:** fichero de configuración de nuestro servlet.
- **web.xml:** donde se encuentra la localización de los archivos de definición del contexto de la aplicación.

Bibliografía

- Arduino, especificaciones:
  http://www.arduino.cc/es/
  http://www.arduino.cc/en/Main/Howto

- Arduino, montajes e información diversa:
  http://www.bricogeek.com/index/cat/13/
  http://www.sparkfun.com
  http://medialab-prado.es/article/viernes_openlab

- Especificaciones de ZigBee:
  http://www.zigbee.org/ZigBeeSpecificationDownloadRequest/tabid/311/Default.aspx

- Esquemático del Xbee Shield para Arduino:
  http://www.arduino.cc/en/uploads/Main/XbeeShieldSchematic.pdf

- Comunicación en Java, uso de puertos serie y USB:
  http://java.sun.com/products/javacomm/

- RXTX, biblioteca Java para comunicación serie y USB:
  http://users.frii.com/jarvi/rxtx/

- Framework Spring, referencia oficial y API:
  http://www.springsource.org/
  http://www.springsource.org/documentation

- Framework Spring, tutoriales:
  http://static.springframework.org/docs/Spring-MVC-step-by-step/
  http://www.roseindia.net/spring/index.shtml

http://maestric.com/en/doc/java/spring

- JApplet en Java, API:
  http://java.sun.com/javase/6/docs/api/javax/swing/JApplet.html

- OpenChart2:
  http://approximatrix.com/products/openchart2

- jQuery, documentación y API:
  http://jquery.com/

- Flot, gráficos en Javascript/jQuery:
  http://code.google.com/p/flot/

- JasperReports, generación de informes:
  http://jasperforge.org/plugins/project/project_home.php?projectname=jasperreports

Los abajo firmantes, matriculados en la asignatura de Sistemas Informáticos de la Facultad de Informática, autorizan a la Universidad Complutense de Madrid (UCM) a difundir y utilizar con fines académicos, no comerciales y mencionando expresamente a sus autores el presente Trabajo de Proyecto Fin de Carrera: "Mountain Bytes: Sistema de Control y Comunicación Inalámbrica de Redes Sensoriales Aplicado al Ejercicio", realizado durante el curso académico 2008-2009 bajo la dirección del Dr. Luis Hernández Yáñez en el Departamento de Ingeniería del Software e Inteligencia Artificial (ISIA), y a la Biblioteca de la UCM a depositarlo en el Archivo Institucional E-Prints Complutense con el objeto de incrementar la difusión, uso e impacto del trabajo en Internet y garantizar su preservación y acceso a largo plazo.

Helena Lorenzo Granizo

Javier Murillo Yagüe

Alberto Velázquez Alonso